CrossTalk

August 2005 **The Journal of Defense Software Engineering** Vol. 18 No. 8

Systems: Fielding Capabilities

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **AUG 2005**	2. REPORT TYPE	3. DATES COVERED **00-00-2005 to 00-00-2005**
4. TITLE AND SUBTITLE **CrossTalk: The Journal of Defense Software Engineering. Volume 18, Number 8, August 2005**		5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **OO-ALC/MASE,6022 Fir Ave,Hill AFB,UT,84056-5820**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**32**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Systems: Fielding Capabilities

4 Key Elements in Fielding Capabilities
These authors discuss how the 76th Software Maintenance Group successfully develops software for the Department of Defense using two key elements: high fidelity test environments and collaboration.
by John D. Holcomb and Michael Hoehn

8 Delivering Capabilities Through Partnerships
Here is how the 402nd Software Maintenance Group successfully partnered with Northrop Grumman Corporation to maintain the complex E-8C Joint Surveillance Tactical Airborne Radar Systems software.
by Chris D. Moore

12 MILS: Architecture for High Assurance Embedded Computing
These authors propose that the Multiple Independent Levels of Security architecture is a solution to meet the needs for critical information assurance systems that deliver the strongest degree of security and safety.
by W. Mark Vanfleet, R. William Beckwith, Dr. Ben Calloni, Jahn A. Luke, Dr. Carol Taylor, and Gordon Uchenick

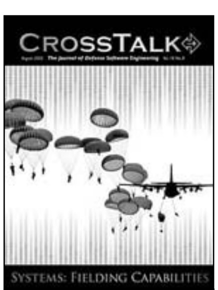
Software Engineering Technology

17 Six Steps to a Successful COTS Implementation
This author explains how a successful implementation of a commercial off-the-shelf-intensive software system can save programs money – if you have the right solution and understand the potential risks involved.
by Arlene F. Minkiewicz

22 Performance-Based Earned Value
Performance-Based Earned Value is a set of principles and guidelines that specify the most effective measures of cost, schedule, and product quality performance, which is distinguished by its focus on customer requirements.
by Paul J. Solomon

Open Forum

27 Balanced Scorecards: From Golf to Business
If you want to be No. 1 in your golf game or in your business, this author shows how effective use of the Balanced Scorecard tool can help get you there by providing balanced measures with supporting data.
by Bill Ravensberg



On The Cover
Cover Design by Kent Bingham.

Departments

3 From the Sponsor
From the Publisher

7 Coming Events

30 Web Sites

31 BackTalk

CrossTalk

OC-ALC/76 SMXG Co-Sponsor	*Kevin Stamey*
OO-ALC/309 SMXG Co-Sponsor	*Randy Hill*
WR-ALC/402 SMXG Co-Sponsor	*Tom Christian*
Publisher	*Tracy Stauder*
Associate Publisher	*Elizabeth Starrett*
Managing Editor	*Pamela Palmer*
Associate Editor	*Chelene Fortier-Lozancich*
Article Coordinator	*Nicole Kentta*
Creative Services Coordinator	*Janna Kay Jensen*
Phone	(801) 775-5555
Fax	(801) 777-8069
E-mail	crosstalk.staff@hill.af.mil
CrossTalk Online	www.stsc.hill.af.mil/crosstalk

Oklahoma City-Air Logistics Center (76 SMXG), Ogden-Air Logistics Center (309 SMXG), and Warner Robins-Air Logistics Center (402 SMXG) SMXG Software Maintenance Groups (SMXG) are the official co-sponsors of CrossTalk, The Journal of Defense Software Engineering. The SMXG's and the Software Technology Support Center (STSC) are working jointly to encourage the engineering development of software to improve the reliability, sustainability, and responsiveness of our warfighting capability.

The STSC is the publisher of CrossTalk, providing both editorial oversight and technical review of the journal.

Subscriptions: Send correspondence concerning subscriptions and changes of address to the following address. You may e-mail us or use the form on p. 30.

309 SMXG/MXDB
6022 Fir AVE
BLDG 1238
Hill AFB, UT 84056-5820
(801) 775-5555

Article Submissions: We welcome articles of interest to the defense software community. Articles must be approved by the CrossTalk editorial board prior to publication. Please follow the Author Guidelines, available at <www.stsc.hill.af.mil/crosstalk/xtlkguid.pdf>. CrossTalk does not pay for submissions. Articles published in CrossTalk remain the property of the authors and may be submitted to other publications.

Coming Events: Please submit conferences, seminars, symposiums, etc. that are of interest to our readers at least 90 days before registration. Mail or e-mail announcements to us.

CrossTalk Online Services: See <www.stsc.hill.af.mil/crosstalk>, call (801) 777-7026, or e-mail <stsc.webmaster@hill.af.mil>.

Back Issues Available: Please phone or e-mail us to see if back issues are available free of charge.

Our Job Is to Get It There

One of the beauties of software is how quickly we can make changes compared to the hardware world. This has given rise to the tremendous expansion of military software applications, which has occurred during the past thirty odd years. I recently saw an illustration that plotted the number of lines of software on U.S. fighter aircraft starting with the F-111 and F-4 and progressing to the F/A-22 and F-35. It was exponential as one might expect (we software types are beneficiaries of Moore's Law and hardware improvements after all). The interesting point to remember is that this embarrassment of riches contains a daunting challenge, namely efficiently harnessing this great flexibility that software affords. We have found over the years that there is "no royal road to learning" as Euclid said. To do software right we have to do it the old-fashioned way, that is, relentless commitment to quality: employing peer reviews, configuration control, documentation, and testing.

The topic of this issue is "Systems: Fielding Capabilities." It is really the reason that all of us who provide software capability are employed. It is our job to get what is needed to those who need to use it. Often we think of software as an intellectual pursuit. All of us have felt the exultation of solving a difficult problem or finding a creative solution but ultimately we must focus on our customer, not ourselves. To this end, it is not enough that we satisfy ourselves with our software abilities but that we satisfy those who depend upon us. Fielding capabilities entails doing our work both well and quickly. Well so the warfighter is not disappointed, or worse, in undue risk, and quickly so the warfighter is not kept waiting. This issue will help you do that.

Thomas F. Christian, Jr.
Warner Robins Air Logistics Center Co-Sponsor

From the Publisher

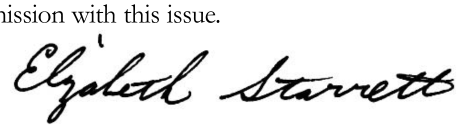
Stay Focused on the User

This month's theme, "Systems: Fielding Capabilities," developed during a discussion with CrossTalk's sponsors about merging hardware and software. As they considered this, they emphasized that importance should not be placed on the system components of hardware and software; importance should be placed on the capability required by the user. This is such an important concept that we decided to build a theme around it.

We start our theme discussion with *Key Elements in Fielding Capabilities* by John D. Holcomb and Michael Hoehn. These authors discuss the benefits of testing environments that emulate the true user environment and the need for ongoing communication among stakeholders. In *Delivering Capabilities Through Partnerships,* Chris D. Moore details his experience with a military-industry partnership focused on providing the U.S. military with needed core competencies to ensure continued support to the warfighter. We conclude our theme articles with *MILS: Architecture for High Assurance Embedded Computing* by several authors who discuss information assurance as a method for enabling the capability to win wars via superior knowledge.

We continue this issue with our supporting articles. In *Six Steps to a Successful COTS Implementation,* Arlene F. Minkiewicz shares key points to contemplate and implement when considering commercial off-the-shelf as part of your end-product. Paul J. Solomon and Bill Ravensberg also contribute with *Performance-Based Earned Value* and *Balanced Scorecards: From Golf to Business*, respectively, explaining how earned value and balanced scorecards are each key to tracking project progress.

CrossTalk continues to aim to provide readers with the capability to buy and build software better. I hope we further our mission with this issue.



Elizabeth Starrett
Associate Publisher

Key Elements in Fielding Capabilities

John D. Holcomb and Michael Hoehn
76th Software Maintenance Group

Organizations that develop software for the Department of Defense must have knowledgeable people to do the work according to documented and mature processes and standards that guide how the work is accomplished. The organization must also have in place the hardware and tools that are used to execute the processes to develop and test the end products. Success of their efforts depends on two key elements: the fidelity of the test environment, and the amount of collaboration with other agencies involved in their program.

The mission of the 76th Software Maintenance Group (SMXG, formerly MAS) at the Oklahoma City-Air Logistics Center, Tinker Air Force Base (AFB), Okla., includes positioning operational capabilities in the field, and improving and adding to them through software development and sustainment. The 76th SMXG performs this mission for the E-3, B-1, B-2, and B-52 aircraft, and for the Air Launched Cruise Missile (ALCM), Conventional Air Launched Cruise Missile (CALCM), and Advanced Cruise Missile (ACM) weapons. The group also has extensive capability for development and maintenance of Test Program Set hardware and software for automatic test equipment; industrial automation; and jet engine testing, trending, and diagnostics.

Any organization that develops or maintains weapon system software must have certain resources in place. These resources include people, a development environment, a test environment, tools, and facilities to house these resources. Policies, instructions, standards, and processes are required to control how the work is accomplished. Measurement and metrics requirements must be established to facilitate tracking workload/labor; to evaluate financial and project performance; and to establish the foundations for pricing future projects, making management decisions, and process improvement efforts. We have an outstanding Software Engineering Process Group (SEPG) that organizes and develops processes and standards, and maintains them online. It also establishes and manages our measurement and metrics requirements and process improvement efforts and the organizational software quality program.

The 76th SMXG consists of approximately 500 engineers, computer scientists, and staff personnel, the majority of whom have in excess of 15 years experience in the organization. Various development environments are used based on the weapon system or automatic test equipment that the application software runs on, however, most applications are developed on IBM mainframes, Sun workstations, or networked personal computers. Tools used include assemblers, compilers, and tools for project planning and management, labor tracking, requirements tracking, configuration management, documentation, etc. A detailed discussion of all aspects of our operation is not possible within the scope of this article, and most readers are aware of these aspects from their own experience. Thus this article will focus on two key areas that reduce risk when fielding operational capabilities: high fidelity test environments and collaboration.

High Fidelity Test Environments

The test environment is one of the key elements in fielding capabilities. If it does not emulate the fielded system to the maximum extent possible, then the risk of operational problems when the software is fielded increases. The initial Avionics Integrated Support Facility Military Construction Project at Tinker AFB was built in the early 1980s to provide floor space to house the software support personnel and development environments for the E-3, B-52, Short Range Attack Missile (SRAM), and the ALCM. An example of our high fidelity test environments, the B-52 Avionics Integrated Support Facility (AISF), is a hot mockup of the aircraft avionics interfaced with the controls and displays. Simulated dynamics of the aircraft are provided by a vehicle system simulator, and weapon simulation is provided by a weapon system simulator.

The SRAM and ALCM laboratory area was built adjacent to the B-52 AISF area. The cruise missile project provided interfaces between the B-52 AISF and empty/loaded pylon/launcher station as well as between the AISF and the ALCM subsystem simulator. This test environment is used for simulation and test of the B-52 operational flight software, the aircraft to missiles interfaces, and the missile operational flight software. By utilizing the combination of the B-52 AISF and pylon or launcher loaded with test missiles, all communication between the aircraft and missiles can be effectively tested. The SRAM program has been disposed of and the missiles laboratory has evolved through the years to include capability for ACM and CALCM.

A missile electronic subsystem simulator, consisting of a table of interfaced missile electronics, is also available. Breakout boxes can be used at the umbilical connector, or at any of the internal missile interface connectors to allow monitoring of the interfaces. Testing of the B-52 operational flight software and missile operational flight program is accomplished by first planning the mission, then flying the aircraft mission on the AISF, rotating the launcher to the proper missile, if necessary, then simulating launch, and finally simulating free flight of the missile to target.

The AISF interface with the ALCM subsystem simulator is used to test aircraft/missile interface up to launch and subsequently test free-flight simulation of the missile operational flight program from launch to detonation at target. Successful testing of B-52 and missile operational flight software and mission planning software in this laboratory provides very high confidence that flight testing will be successful and that the software will provide the required capability when fielded.

A government owned and operated test environment for weapon systems has benefits other than the ability to fully test the software. Having this capability in a government facility allows it to be used for competitive procurement of projects that are beyond organic capability. For example, a B-52 modification was programmed and funded, but the sole-source contractor's price for developing the modification at the company's facility was in excess of the budget. We recommended to the program office that the project be competitively procured based on performance in our government facility. The effort was competed, development and testing was accomplished in the AISF by the winning contractor, and the final cost was approximately one half of the original bid.

Another additional benefit is the expertise that organic personnel develop as a result of having this type of laboratory. An example of this occurred after two B-52/ALCM-W80-1 Joint Test Assembly (JTA) flight tests resulted in aborted launches with total mission failure. An analysis of the mission data indicated a problem between the B-52 offensive avionics system (OAS) and the missile test payload (W80-1 JTA). After returning to the home base, the aircraft underwent extensive ground testing by Air Force personnel and no problem could be found. Sandia National Laboratories performed a series of comprehensive tests on their JTA package and concluded that it did not contribute to the aborted launches. Sandia further prepared a letter to the Cruise Missile Product Division detailing their findings and recommending the Air Force suspend future B-52/ALCM JTA flight tests until the Air Force could identify the source and correct the problem.

This problem was referred to our engineers, and the B-52/missiles laboratories were configured for an ALCM JTA launch using missile and B-52 production hardware and operational software. Utilizing state-of-the-art recording and analysis tools, engineers performed multiple JTA launches. Analysis of the laboratory flight test data and the JTA data from the aborted launches clearly determined that the aborted launches were a result of the JTA package. Having determined the source of the problem, engineers from the AISF presented their findings identifying the JTA as the source of the problem and isolating specific circuits in the JTA that were suspect. Sandia accepted these findings, had additional testing performed on the suspected circuits, and was able to isolate the specific failure mode.

Similar test environment capabilities exist in all of our weapon system laboratories. For example, the E-3 Airborne Warning and Control System (AWACS) laboratory has both surveillance radar configurations; this way, all versions of the E-3 software can be tested. During Desert Shield/Desert Storm, an enhancement was required to support that effort. The requirement was identified on Thursday. The change was programmed, implemented, tested in the laboratory, flight tested at Tinker AFB, and sent to the theatre on a resupply flight the next Monday. This demonstrated the fast turnaround capabilities of organic resources with high fidelity test environments.

This system has also helped with software not developed by our organization. During the early 1980s, the AWACS wing experienced a serious problem with the E-3 navigational computer system. The system consistently failed to capture the turn portion of an established surveillance orbit, potentially causing the E-3 to fly into unauthorized airspace. Serious consequences were narrowly avoided on several occasions. Once notified of the problem, the E-3 AISF was able to reproduce the problem in the laboratory, locate the source in another organization's code, and develop a fix. The modified operational flight program was then tested in the E-3 AISF and delivered to the E-3 fleet in a timely manner.

These types of weapons system test environments are normally established as a part of the weapon system development program at the prime contractor's facility. The test equipment is then either duplicated at or transferred to a government facility for support of the weapon system software after deployment of the weapon system. Since the test environment includes the avionics suite, the cost is very high. In 1995, replacement cost estimates for the AISFs were as follows:

- B-1: $171 million.
- B-52: $54 million.
- Missiles: $51 million.
- E-3: $100 million.

Each AISF is unique and may have more than one hot mockup of the avionics. Simulation computers are typically Harris (now Concurrent) computers and use Fortran, C, or C++, as the source language(s) for the simulation software depending on the specific application. Our experience is that the high cost of these high fidelity test environments is well worth the investment because they enable the software support activity to provide the customer and the warfighter with the required capabilities when they are needed.

Collaboration

Another key element in fielding capabilities is teamwork between the program manager, system engineer, software developer, warfighter, and tester. The B-52 Mission Planning Software Section is one of our top success stories. This section has responsibility for development and sustainment of the B-52 mission planning software (B-52 Aircraft/Weapons/Electronics [A/W/E]) that runs on the Air Force mission support system (AFMSS). This system essentially automates the process that the flight crews previously performed manually – according to the Technical Orders (TOs), which are used as the basis for the software requirements. One of the most important keys to success is customer involvement. To ensure that the product produced meets the user/customer requirements, the warfighters are involved in each phase throughout the development.

The B-52 A/W/E is one element of the complete mission planning environment, which is a combination of 35 different elements of software and 17 pieces of dedicated B-52 aircraft software, as shown in Figure 1 (see page 6); the Glossary defines the terms in the figure. These are developed by different agencies and contractors, and many serve multiple weapon systems. Integration of all of these applications on a single system to meet overall warfighter requirements is a significant part of our effort.

The main key to success in this area is the in-depth understanding of the entire environment, both hardware and software. The mission planning process is tested from beginning to end, including the production of all flight products. These products are taken through the final verification of actually loading them in our B-52 AISF and flying the missions, complete with weapons, and the recording of all the data for analysis.

Ensuring success begins with customer or user relationships. More than a third of our key people who develop and maintain mission planning applications are on a first-name basis with dozens of stakeholders:

- **B-52 System Program Office**. All mission planning system engineers, program managers, and weapon system integration engineers communicate several times a week.
- **46th Operations Group/Test Squadron**. Responsible Test Organization for mission planning; participates in our development test (DT), DT/operational test (OT), and formal qualification test (FQT).
- **28th Test Squadron**. Final test authority for force development evaluation (FDE).
- **Air Force Operational Test and Evaluation Center (AFOTEC) Detachment 2**. Official OT agency for mission planning.
- **AFOTEC Detachment 5**. Official OT agency for weapon systems like the Cruise Missiles and Joint Air to Surface Stand-off Missile (JASSM).
- **49th Flight Test Squadron**. B-52 Flight Test Squadron at Barksdale AFB.
- **5th Operational Support Squadron**. Warfighters from Minot AFB.
- **2nd Operational Support Squadron**. Warfighters from Barksdale AFB.
- **Mission Planning System Support Facility**. Air Force mission planning software integration, distribution, and support from Hill AFB.

As noted above, throughout the applica-

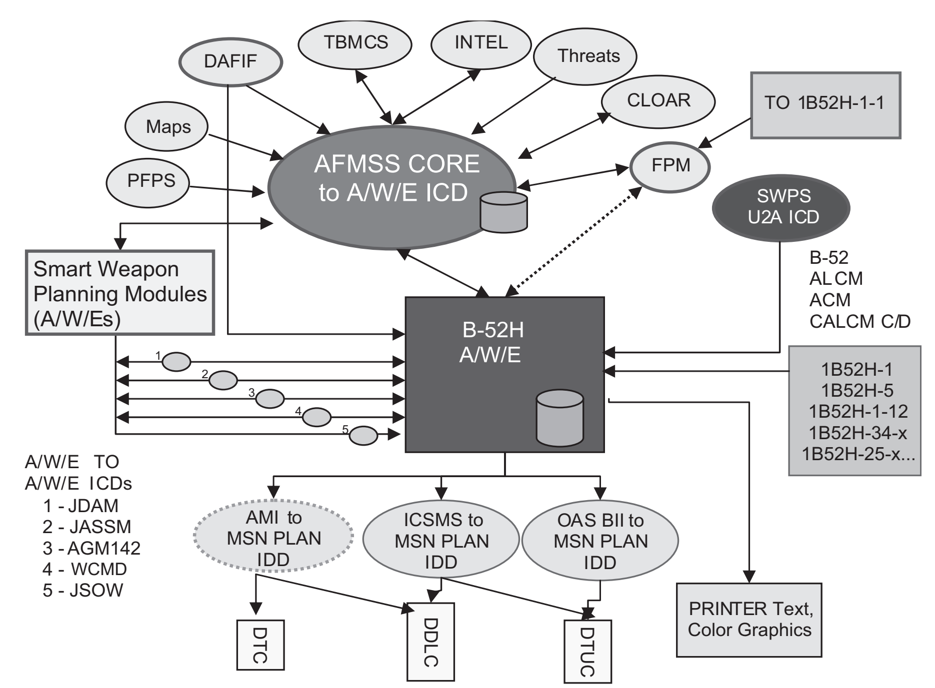
Glossary

A/W/E	Aircraft/Weapons/Electronics
ACM	Advanced Cruise Missile
AFMSS	Air Force Mission Support System
AFOTEC	Air Force Operational Test and Evaluation Center
AFPD	Air Force Policy Directive
AGM	Air-to-Ground Missile
AISF	Avionics Integrated Support Facility
ALCM	Air Launched Cruise Missile
AMI	Avionics Midlife Improvement
AWACS	Airborne Warning and Control System
CALCM	Conventional Air Launched Cruise Missile
CALCM C/D	Two versions of the CALCM
CLOAR	Common Low Observability Auto Router
DAFIF	Digital Aeronautical Flight Information File
DDLC	Digital Data Loader Cartridge
DT	Development Test
DTC	Data Transfer Cartridge
DTUC	Data Transfer Unit Cartridge
FDE	Force Development Evaluation
FPM	Flight Performance Model
FQT	Formal Qualification Test
ICD	Interface Control Document
ICSMS	Integrated Conventional Stores Management System
IDD	Interface Definition Document
INTEL	Intelligence Data
JASSM	Joint Air-to-Surface Standoff Missile
JDAM	Joint Direct Attack Munitions
JSOW	Joint Standoff Weapon
JTA	Joint Test Assembly
OAS	Offensive Avionics System
OT	Operational Test
PFPS	Personal Flight Planning System
SRAM	Short Range Attack Missile
SEPG	Software Engineering Process Group
SMXG	Software Maintenance Group
SWPS	Strategic War Planning System
TO	Technical Order
TBMCS	Theater Battle Management Core System
U2A	U.S. Strategic Command to AFMSS
WCMD	Wind Corrected Munitions Dispenser

tion's life cycle the stakeholders meet frequently via face-to-face meetings and teleconferences. At least once a year, a mission planning open house is conducted at the user's base of operations. This includes several days for familiarization with our existing applications and user interface working group meetings to discuss upcoming designs, priorities, trade-offs, and requirements. Our engineers and computer scientists also hit the road, spending an average of more than 200 man-days per year in the field, meeting with the users, other developers, and other program offices to continually coordinate efforts, schedules, and requirements, and to provide familiarization with our systems.

Requirements review boards for our A/W/E applications and AFMSS core are conducted and defect review boards are held following each formal test. Eight official test events were hosted last year. Each event had one or more users or customers working side-by-side with the developers. As part of the development effort, several DTs are hosted prior to the FQT. At least one of these DT tests will be a combined DT/OT that is a development test using operational procedures, data, and crew members, making it as real-world as possible. This process is a profound strength in the organization. Feedback is received from the users continually throughout the development phase.

Figure 1: *B-52 Mission Planning Environment*



The OT certification brief is prepared and presented. Using Air Force Manual 63-119, Certification of System Readiness for Dedicated Operational Test and Evaluation, a matrix of 33 certification templates is evaluated that identifies specific problem or risk areas that could hinder the smooth transition from development, through test, to the fielding of a product. All of the communities listed above participate in this process, and the entire group agrees that the product is ready to be tested. This final step provides complete confidence that the products will meet the warfighter's expectations the first time, every time. When a product approaches fielding certification, the actual users have seen it, used it, evaluated it, tested it, and stand behind it along with the developers.

A recent success story centers around a flight performance change to the way mission planners need to account for drag on the B-52 because of external weapons hanging on the wings. The multiple weapon configurations create different fuel burn rates affecting the range of the aircraft. Implementing this change in the TO spanned three software releases and required participation from almost every organization listed above.

Through requirements review, software development, integration, test, and the defect review board process, each spiral would further refine the requirements, each time giving the user increased capabilities and enhancing the system effectiveness to plan aircraft missions. It became clear early when dealing with this issue that each member of the team had a different piece of the puzzle. Only through continuous communication and collaboration were the answers to all the questions understood enough to produce a quality tool for the warfighter.

From start to finish, nothing is done in a vacuum without the user. A lot of companies will offer the user an early look or attendance at design reviews, but the customer seldom gets the complete picture or real hands-on experience during development of its system. As explained here, Tinker AFB's B-52 mission planning section goes above and beyond to get the user involved in every step. Nothing is hidden or kept from the customer. By involving the users in requirements reviews, early DT events, and using our integrated suite of test facilities, the customer is allowed to actually run the system end-to-end in one location. A demonstration of one or two isolated pieces of the puzzle is not needed because the user sees the whole enterprise and walks away with a feeling of confidence that what is paid for will provide the capabilities required in the field.

This type of team effort is becoming more important with the Air Force Policy Directive (AFPD) 63-1 cited commander's intent that states, "The primary mission of our acquisition system is to rapidly deliver to the warfighter affordable, sustainable capability that meets their expectations." The objective of AFPD 63-1 is to "create a context that allows the program manager to shape and execute a program with an emphasis on teamwork, trust, common sense, and agility." It further states that "the warfighters, developers/acquirers, technologists, testers, budgeters, sustainers, and industry must plan and execute together in order to meet the Commander's intent." These seem to be lofty ideals, but we have proven they can be done.

Summary

Fielding capabilities can be enhanced by having high fidelity test environments and by collaboration between all of the participants on a program. The Air Logistics Centers at Robins AFB and Hill AFB have similar capabilities to those that are described in this article, although for different weapon systems. Program managers may be able to take advantage of existing organic resources to reduce cost and risk. Further, partnering agreements can be established between organic software support activities and contractors to facilitate utilization of organic resources in teaming arrangements to work jointly on Department of Defense projects. All three Air Logistic Center software support activities have Web sites (see page 30) that provide contacts.◆

About the Authors

John D. Holcomb is the Operational Flight Programs technical expert in the 76th Software Maintenance Group at the Oklahoma City-Air Logistics Center, Tinker Air Force Base, Okla. He has 40 years of federal service, including 33 years of experience in weapon systems software, and participated in the development of the Department of Defense Standard 2167. He has a bachelor's degree in math and physics (double major) from the University of Central Oklahoma.

76th SMXG
4750 Staff DR
Tinker AFB, OK 73145-3318
Phone: (405) 736-3835
Fax: (405) 736-3584
E-mail: john.holcomb@tinker.af.mil

Michael Hoehn currently manages the B-52 Mission Planning Software Section in the 76th Software Maintenance Group at Tinker Air Force Base, Okla. He has more than 15 years experience in both Test Program Set and Operational Flight Program software development and maintenance.

76th SMXG
8080 Perimeter RD STE 107
Tinker AFB, OK 73145
Phone: (405) 736-5539
Fax: (405) 736-4129
E-mail: michael.hoehn@tinker.af.mil

Delivering Capabilities Through Partnerships

Chris D. Moore
Warner Robins-Air Logistics Center

As public-private partnerships become more prevalent in the Department of Defense for providing logistics support for advanced weapon systems, integrated teams must look for unconventional opportunities to exploit the best capabilities of their combined resources to support many diverse program objectives. The challenge is figuring out how to evolve traditional customer-supplier relationships into truly integrated teams with common objectives at the forefront. Warner Robins-Air Logistics Center and the Northrop Grumman Corporation are pioneering a new path with their partnered E-8C Joint STARS software maintenance team.

The 402d Software Maintenance Group (SMXG, formerly MAS) at Warner Robins-Air Logistics Center (WR-ALC), Robins Air Force Base (AFB), Ga., is soaring high in its partnership with Northrop Grumman Systems Corporation (NGSC) to maintain the complex E-8C Joint Surveillance Target Attack Radar System (Joint STARS) software through integrated teaming. As legislative support and expectation for government-private partnerships continue to grow, software organizations are finding more opportunities to leverage the best of each side's infrastructure to provide both government core capability and optimum support for fielded systems.

The challenge is figuring out how to optimize the new and unconventional government-private relationship. While many government-private partnerships cast the parties in traditional contractor-subcontractor roles, the 402 SMXG and NGSC partnership has taken the concept to a new level with a truly integrated team fielding war-winning capabilities through software maintenance.

The E-8C Joint STARS Systems Group (E8SG) is a long-range air-to-ground surveillance system designed to locate, classify, and track ground targets in all weather conditions. It is operated by the 116th Air Control Wing (ACW) based at Robins AFB. In 1998, the Air Force designated Joint STARS a core workload. The Department of Defense (DoD) services designate certain weapon systems, equipment, and components as mission-essential for support of scenarios approved by the Joint Chiefs of Staff. The DoD ensures that there is core depot maintenance capability to support these mission-essential weapon systems. Core exists to minimize operational risks and to guarantee required readiness for these weapon systems.

In November 2000, the 330 Intelligence Reconaissance and Surveillance Group (330 IRSG), also at WR-ALC, awarded a Total System Support Responsibility (TSSR) contract to NGSC, which was the Joint STARS system integrator operating out of its Airborne Ground Surveillance and Battle Management Systems Division in Melbourne, Fla. This contract made NGSC responsible for maintaining and providing logistics support for the entire fleet of 17 E-8C jets from nose to tail. In that Joint STARS is a core logistics workload, a partnership was required between WR-ALC and NGSC to enable WR-ALC to support NGSC's logistics responsibilities with government-furnished supplies and services (GFS/S). While this partnership includes the support of many areas of logistics, this article focuses on Joint STARS software maintenance.

> *"Success hinges on the deliberate support of all objectives by all parties."*

The fiscal year 1998 Defense Authorization Act provided statutory authority for DoD depots to enter into government-private cooperative arrangements (partnerships) for the performance of depot-level work. While the concept of partnering or teaming with industry was conceived initially as a strategy for depots to make skills, facilities, and equipment available to the private sector to perform government workload and to maximize the utilization of skills, facilities, and equipment that are required to support core workloads, it lent itself well to a somewhat different arrangement regarding Joint STARS software maintenance. In this case the contractor, NGSC, had a history of maintaining Joint STARS software almost exclusively. Further, the facilities and expertise existed at the contractor facility in Melbourne, not at Robins AFB. In this case, the government is the novice and the contractor is the expert.

As an umbrella to the WR-ALC-NGSC partnership, a Long Range Memorandum of Agreement (LRMOA) was established between the three principle parties depicted in Figure 1. They are WR-ALC, specifically the 402d Maintenance Wing (MXW), formerly WR-ALC/Maintenance Directorate; the E8 Systems Group, formerly the Joint STARS System Program Office at Electronics System Command (ESC); and NGSC.

The agreement set the ground rules for partnering by acknowledging the existence of both common and unique individual objectives. While there are mutual objectives in government-private partnerships, there are many objectives that are unique to the parties and may have higher priorities to the respective parties.

Success hinges on the deliberate support of all objectives by all parties. The challenge is pioneering innovative methods for achieving these objectives that sometimes appear to be in complete conflict. The agreement recognized the new and unique roles of the parties as well as individual objectives, but most importantly set forth the agreement to pursue a set of common objectives.

The system program director E8SG objectives are to implement U.S. Air Force (USAF) program management and acquisition techniques as necessary to provide the best possible support to the warfighter while ensuring that best value is achieved for the USAF. The focus of E8SG is on providing the user with superior combat capability by balancing program costs, schedule, and performance elements.

The WR-ALC objectives are, pursuant to law, to maintain the appropriate levels of core competencies in logistics management, engineering, supply chain management, and depot-level maintenance necessary to ensure efficient and cost-effective sustainment for all of its assigned weapon systems now and in the future. To achieve these objectives, the center must develop in-house expertise and transition software maintenance workload from the contractor to organic while mitigating risk to the user and to the overall program. Success in this endeavor satisfies the core directive.

An additional WR-ALC objective is to partner with the contractor to reduce the core capital investment through sharing of equipment and resources.

Contracting Objectives

As the TSSR contractor, NGSC objectives are to fulfill an important role in assisting the USAF organizations in achieving their respective objectives. Consequently, NGSC's primary objective is to ensure the Joint STARS weapon system continues to provide the user with superior and reliable combat capability. This performance, in turn, allows NGSC to perform its TSSR contract obligations. By providing superior Joint STARS support, NGSC establishes a positive reputation for the Joint STARS weapon system, earns maximum performance incentives, and enhances its reputation as a leading provider of Joint STARS support.

When these individual objectives are superimposed, a set of common objectives comes into focus. The parties' intent is that through this LRMOA, and other subordinate agreements, the parties will successfully integrate their efforts to continuously improve total systems support of Joint STARS. Notwithstanding the unique roles and responsibilities of each independent party, the parties collectively agree that their overarching mutual objectives are to provide the following:

- Superior support to the warfighter.
- Best value to the USAF (balancing both program and broader objectives).
- The mechanisms necessary to meet the USAF core logistics competencies requirements.
- Support to future core and source-of-repair assignment analyses and decision-making to enable the USAF to balance objectives of the Joint STARS program with broader USAF imperatives such as the maintenance of core competencies.
- The creation of an integrated digital environment (IDE) as a key enabler in achieving the communication, coordination, insight, and responsiveness objectives outlined in this document (discussed later).
- The ability of NGSC to achieve reasonable profits and enhance its corporate reputation through demonstrated performance in achieving the aforementioned common objectives as applicable to the TSSR contract.

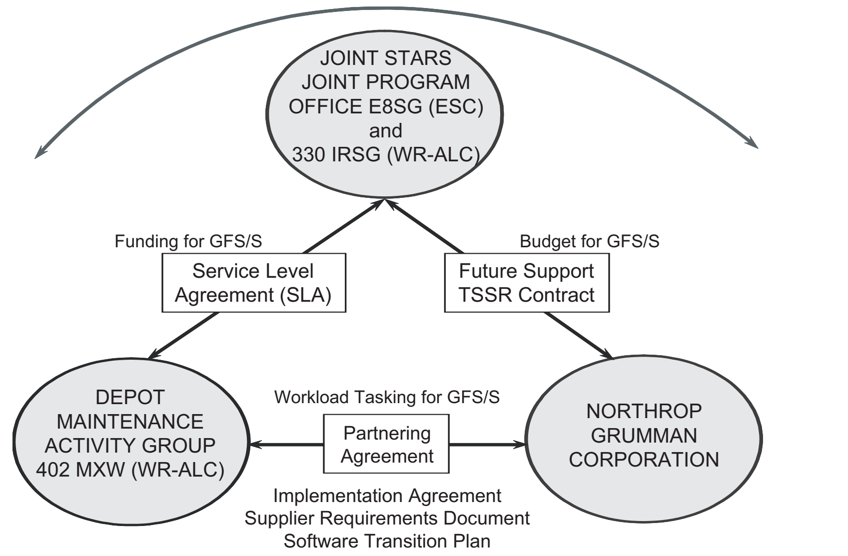


Figure 1: *Long Range Memorandum of Agreement*

Clearly there are many objectives ripe for competing interest. The TSSR contractor must balance maximizing corporate performance with supporting the Air Force in meeting its core capability requirements. The government partner must acquire laboratories, personnel, and training, and develop expertise and transition workload while mitigating program risks. It is a tough challenge, but major risk mitigators include early identification and optimization of individual strengths, practical integration of engineering and management processes and human resources, implementation and continued optimization of communication processes and procedures, and identification of weak areas followed by development of tactical plans to effectively fill those voids. Intense focus in these areas with little regard for organizational boundaries and contractor/government affiliations produces not only a partnered team, but also an integrated team postured to improve efficiency, performance, and quality – a win-win scenario. The bottom line: all objectives are met and, most importantly, superior warfighter support is provided.

There is also the concept of trust. Traditionally the government personnel have been perceived as the customer in all instances. The partnership scenario casts the government partner in an entirely different role: an insider, a colleague, and yes, in some forms even as a subordinate. Not only is this scenario foreign to the contractor, but extremely new and possibly uncomfortable for the government personnel. It takes deliberate attention and actions to bridge this gap.

The contractor developed the E-8C and remains fiercely loyal to it, coveting it and its caretaking. In many cases, it is the lifeblood of the company or department and, to some extent, its surrounding community. Eradicating the culture of competition is essential to the success of the government-private partnership, but this cannot be mandated or accomplished externally. It can only come from within the integrated team through building the spirit of teamwork and shared goals as reported in the following section.

Working Together

While many obstacles can be anticipated at the outset and contingencies devised, most lie in waiting just around the corner and can initially appear to be insurmountable. To name a few, there is geographical separation, differences in software laboratory setup and capability, different technical and management processes, risk to contractor performance due to the government's poor performance, competition versus teamwork, and the previously discussed core objectives versus best value.

The E-8C is essentially a spiral development system with continuous development of new system-level enhancements and functionality, and subsequent fielding of these new capabilities through integration into the software maintenance process. Concurrent with new development is ongoing identification of software change requirements to address deficiencies and defects, which is the customary software sustainment life cycle.

The first four years of the partnership took a natural course with respect to software workload sharing and transition. The 402 SMXG team had experience in performing limited modifications to the Joint STARS software specifically in the area of ground support software tools. These tools varied from pre- and post-mission support tools to software development and test tools. At first, NGSC tasked the 402 SMXG to perform software change projects in these areas. As the system integrator prior to the partnership, NGSC integrated – for a fee – any organically produced software modifications, thereby limiting organic contributions to the software maintenance effort.

Under the TSSR contract and through the partnership, NGSC engineering management opened the doors to the NGSC software lab and invited the Robins software engineers to deliver and integrate their own software changes. This single activity laid the foundation for what would become the standard software integration process. The 402 SMXG engineers would become responsible for following NGSC software engineering processes for in-house software delivery, integration, and testing. It also was the first step toward building confidence by the TSSR contractor in the government software maintenance team.

Prior to the partnership and the TSSR contract, software requirements were flowed to NGSC in a very structured and formal process of requirements management by the operational user, the 116th ACW and the 330 IRSG. Under the TSSR contract, NGSC took on a more programmatic role and the responsibility for making capability- and value-based decisions with regard to requirements management. NGSC established several requirements working groups in specific functional areas of the software. These working groups consisted of representatives from all stakeholders, including the user, contractor, government team, and program office.

The working groups are responsible for evaluating software problems, alternative solutions, and ultimately recommending requirements. Involvement by the government software team in these forums is essential to establishing a core software maintenance capability and has proven invaluable to the government team. Final requirements are managed in Integrated Release System Engineering Management Teams (IR SEMT). While this forum is the official communication between the contractor and the program office for software requirements, the 402 SMXG is a participant as well.

Probably as significant as anything has been the inclusion of the 402 SMXG in day-to-day software maintenance team meetings. These vary in function, but are numerous, and all are critical to continuous and optimum performance by the integrated software maintenance team. There are weekly Administrative SEMT meetings where project status and risks are discussed and short-term direction is provided. The weekly Technical SEMT is chaired by the Joint STARS software chief architect. In these forums, engineering leads and subject-matter experts discuss current tasks and issues from a system perspective, and technical guidance is cultivated and provided. In the weekly software integrated product team (IPT) meetings, software change designs are reviewed by leads from each system area to determine correctness prior to approval at the weekly software configuration control board (CCB). In the CCB, completed work is approved for incorporation into the weekly software build, and new assignments are made based on mature requirements and software change requests.

As you can see, the integrated team's concept of operation for a particular software release is one of continuous problem analysis, software modification, integration, and test. Major requirements that flow from the IR SEMT are developed and integrated simultaneously with the continuous correction of defects. Major requirements dictate the overall integrated release schedule and determine when significant modification curtails.

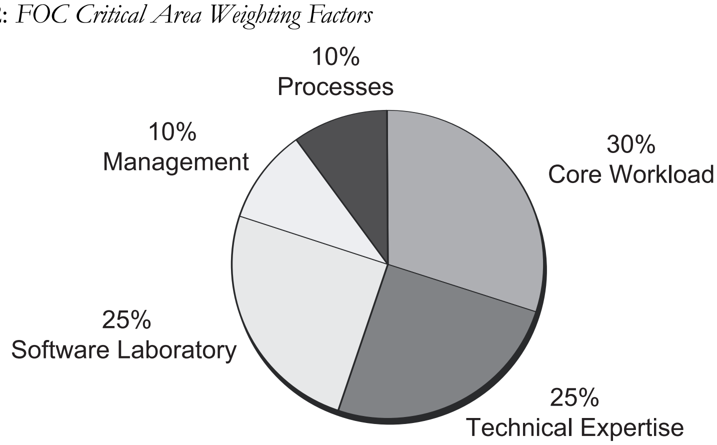
While NGSC has primary responsibility for conducting system-level testing on the E-8C, the 402 SMXG participates in this area as well to further the development of government expertise and capability. In this phase, NGSC must balance mentoring with meeting system requirements and schedule. The 402 SMXG has performed various roles ranging from mission planning using the ground software tools to actually flying on test aircraft and performing system-level tests.

The 402 SMXG possesses an extraordinary benefit in being co-located with the single Joint STARS operational wing, the 116th ACW. Not only does the 402 SMXG reside in the same building as the 116th Computer Systems Squadron that performs pre- and post-mission operations as well as software requirements management, but the 402 SMXG also operates and maintains the Joint STARS mission crew training systems in the same facility. Access to this critical resource enables the 402 SMXG to test the IR's weekly developmental builds on a target system. This can be done because Joint STARS incorporates the concept of a single software baseline. The same software that executes on the E-8C also executes on the trainers. Further, the 402 SMXG has the responsibility of ensuring the IR will operate according to specification on the training system once complete.

Achieving Full Operational Capability

As stated before, the 402 SMXG has benefited immensely by partnering with the Joint STARS integrator and being allowed to perform as an integrated teammate rather than as a subcontractor. This natural teaming process has enabled processes and relationships to form from the ground up rather than be dictated from above. However, in the fourth year it became obvious that a quantitative structure was needed to evaluate when the 402 SMXG would be certified as capable of performing the software maintenance workload; in other words, achieving full operational capability (FOC). A Core Activation Plan (CAP) was established to provide the FOC plan and criteria. The FOC methodology consists of a scoring mechanism to evaluate the government software maintenance

Figure 2: *FOC Critical Area Weighting Factors*



capability against five critically weighted areas as depicted in Figure 2. Those areas include 1) core workload, 2) technical expertise, 3) software laboratory, 4) management, and 5) processes.

This status against this CAP is presented to WR-ALC, NGSC, SPO, and the 116th ACW senior leadership semiannually in the Joint STARS Partnership Review Committee briefing. Upon performing the initial FOC evaluation, we learned that the area needing the most attention was technical expertise. Mentoring would be required from the contractor to the government team in specific areas that are stable with minimal modification, and in specific areas where the government team lacked sufficient experience and had performed minimal or no software changes.

To complete the picture, deliberate steps were required. NGSC increased the level of software maintenance workload assigned to the 402 SMXG in these areas and increased the level and quality of mentoring of government engineers. This initiative ensures quality software is delivered on time in addition to transferring much needed knowledge to the 402 SMXG – again a win-win approach through partnering.

A final, and critically important common objective, was the establishment of an IDE. This goal was conceived at partnership inception, but has been the most illusive. The IDE, as depicted in Figure 3, was planned to be an electronic collaborative workspace to facilitate joint and simultaneous accomplishment of software engineering at geographically separated sites manned by separate teams. For it to become a reality, the IDE required software equipment laboratory upgrades on both ends, security approvals, integration and testing, and concepts of operation.

During the first four years of the partnership, we relied upon weekly overnight shipments between Robins AFB and Melbourne to facilitate the delivery of software products to and from the two sites. In January 2005, our IDE was completed with official operation of a network connection between the two sites. We will use the connection to transfer software files back and forth to facilitate integrated software maintenance teaming, and also to connect remotely to each site for day-to-day operations. We are one step closer to becoming a virtual integrated team via the completion of this connection.

Conclusion

Having said this, let us take a look at our current status. We have integrated

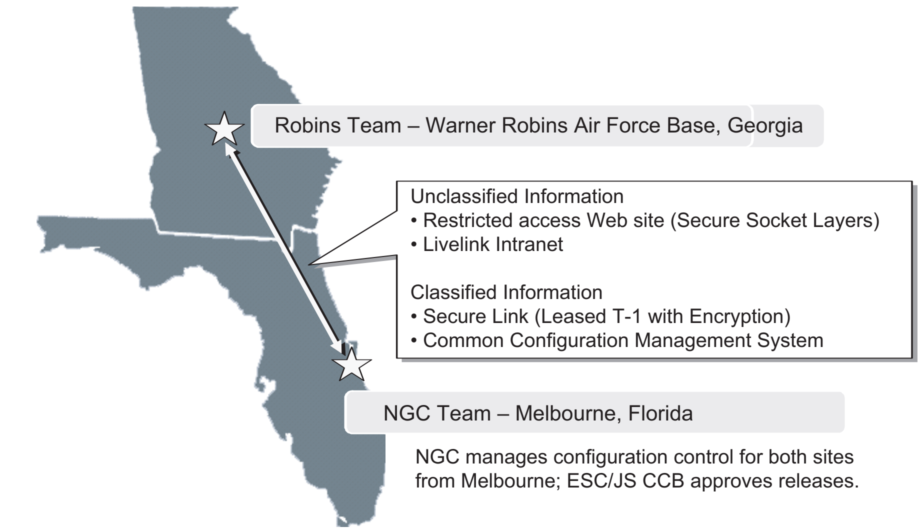


Figure 3: *The Robins AFB and NGC Team Initial IDE*

processes and project management, we have devised a method of sharing software workload, we have made plans to shore up the government's weak areas in system expertise, and we have completed the initial IDE. The 402 SMXG's current score against the FOC criteria established in the CAP is 95 percent out of a required 90 percent. Upon completion of the software release currently in development, the partners plan to declare the government capability fully operational.

Where do we go from here? While the initial period of the partnership has focused heavily on establishing organic software maintenance capability concurrently with satisfying user requirements, the future likely holds in store shrinking budgets and uncertainty. The partnered team will need to increase attention on value and optimization on both ends. With the next generation Joint STARS – the E-10 – on the distant horizon, the current platform will likely continue to undergo upgrades with the addition of new functionality to maintain pace with the demands of global conflict.

NGSC will be called upon to develop new capabilities that exploit the system for maximum operational effectiveness. The 402 SMXG will focus on increasing knowledge base and efficiency to keep up with the flow of software maintenance requirements. Both teams will look for opportunities to reduce costs and optimize processes. Exploiting the IDE will be a central focus for the near term. Joint efforts to strengthen and optimize interfaces with the user to improve software maintenance requirements management will also be a central focus of the partnership.

Sharing and applying lessons learned to other partnered programs within the Air Force Materiel Command will be a significant goal for the coming years. While the future was initially somewhat blurry, the Joint STARS software maintenance partners have exploited their newfound relationship to bring world-class war fighting capability and value for the warfighter into focus.◆

About the Author

Chris D. Moore is the director of the 402d Air Combat Operational Flight Program (OFP) Squadron at Warner Robins-Air Logistics Center, Robins Air Force Base, Ga. He is responsible for sustainment of both the E-8C Joint Surveillance Tactical Airborne Radar System and the F-15 Eagle OFPs. He has 23 years experience in software engineering at Robins ranging from acquiring, developing, and sustaining software for automatic test equipment to sustaining OFP software. Moore has extensive experience with the Capability Maturity Model® Integration and in managing organizations steeped in software process improvement culture. He has a bachelor's degree in electrical engineering and a Master of Science in engineering management.

420 Richard Ray BLVD STE 100
Robins AFB, GA 31098
Phone: (478) 926-2754
Fax: (478) 926-3169
E-mail:chris.moore@robins.af.mil

® Capability Maturity Model is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

MILS: Architecture for High-Assurance Embedded Computing

W. Mark Vanfleet
National Security Agency

R. William Beckwith
Objective Interface Systems

Dr. Ben Calloni
Lockheed Martin Aeronautics Company

Jahn A. Luke
Air Force Research Laboratory

Dr. Carol Taylor
University of Idaho

Gordon Uchenick
Objective Interface Systems

The Department of Defense's increasing dependence on new technology such as unmanned aerial vehicles has created a need for high-assurance systems that deliver the strongest degree of security and safety. However, there has been a notable lack of systems, commercial or government-sponsored research, and engineering that meet these requirements. The Multiple Independent Levels of Security (MILS) architecture is proposed as a solution to meet the needs for critical information assurance. MILS is a componentized architecture based on a commercial off-the-shelf separation kernel that enforces strict communication and partitioned process execution. MILS supports multiple levels of security communication, security policy composition, and modular design so that critical components are able to be evaluated at the highest levels to ensure secure and safe operation.

Information supremacy wins wars. Warfare has always required sharing the right information with the right person at the right time. Technology today enables information sharing on a scale well beyond what our forefathers imagined, but sharing information with the wrong individuals can have catastrophic consequences.

Secure information sharing is critical to enable and protect the warfighter without compromising the mission. The challenge is that warfighter-crucial information is highly diverse. Initiatives such as Network-Centric Warfare, System of Systems, and the Global Information Grid strengthen the desire to share information with multiple levels of security (MLS). MLS systems have historically been among the most challenging and expensive systems to develop and deploy [1]. Sharing and separating information in coalition force operations is an equally challenging and further complicating problem.

Multiple Independent Levels of Security (MILS) is an architecture that makes development, accreditation, and deployment of MLS-capable systems more practical, achievable, and affordable. The MILS architecture significantly increases protection, reduces time to develop, and reduces schedule risk of deploying technology to provide high-assurance systems that are both safe and secure [1].

While the MILS architecture allows for a system of highly secure distributed components, it does not automatically guarantee a secure composed system out of independent secure components. System-wide security is still up to the system designer with MILS providing the building blocks and tools needed to construct a system-level security policy that can then be verified. There are tools, e.g., Boundary Flow Modeling [2], for assuring that security policies compose for a given system.

Where We Have Been

In almost all commercial off-the-shelf (COTS) operating systems and communications technologies, security is an afterthought, addressed via a *fail-first*, *patch-later* paradigm. When a system is penetrated, fixes are then pursued to plug the hole. After a new virus propagates, the enabling weakness is repaired to stop further infection. The frail approach of attempting repair of infected systems is also common. Damage is frequently not detectable or repairable in systems with weak security foundations. Fail-first, patch-later is inappropriate for any mission-critical system because it is reactive and always one step behind the attacker. In mission-critical systems, damage must be avoided or bounded when impossible to avoid. Proactive measures are required to safeguard information and the warfighter, and prevent the damage from happening in the first place.

In traditional architectures, there were good reasons to assign all policy enforcement to a monolithic security kernel. To ensure that enforcement was non-bypassable, security functions had to be part of every system service request. To ensure that enforcement was tamper-proof, security functions had to be in an address space separate from the application [3]. These security functions needed to be in a monolithic security kernel since the computing power of two decades ago was not sufficient to perform the context switches required to separate all of these processes and data and still maintain system performance.

The security kernel with its set of trusted security functions often produced large, complex, unstructured programs that were difficult to certify at the higher assurance levels [4]. SCOMP, which managed to achieve the highest A1 security rating via the historic "Orange Book" [5][1], was based on a very simple security kernel [3]. Most security kernel-based systems never achieved the highest level of certification, which required formal verification. The Motorola Network Encryption System that handled MLS data through encryption achieved a much lower B2 rating, and the XTS-300, the successor to SCOMP, was certified at a B3 rating [3].

Aside from the difficulty of certifying systems with complex, monolithic kernels, the more important problem is in trying to enforce a single, system-wide security policy. For example, Blacker, which successfully handled encryption of MLS data, could not successfully accommodate administrative traffic within its model of classification levels [3]. In general, security policies in the kernel did not provide the robustness required for many applications where application-specific security policies would have provided more tightly focused protection.

Where We Are Going

MILS was created to enable application-level security engineering at a high level of assurance while being affordable. MILS takes advantage of Moore's Law's [6] performance increases over the last two decades by layering small, formally modeled and mathematically verified components together to create a high-assurance foundation. In MILS, applications are empowered to enforce their own security policies instead of relying on generalized kernel security services. MILS also enables efficient systems engineering, where high-assurance components can be effectively reused without modification. This lowers certification costs since certification artifacts can be reused.

The concept of MILS originated in papers written by John Rushby, Ph.D, of the Stanford Research Institute in the early 1980s [7, 8]. Rushby proposed that a separation kernel divide memory into partitions using the hardware memory management unit and allow only carefully controlled communications between non-kernel partitions. This allows one partition to provide a service to another with minimal intervention from the kernel [7].

Traditional operating system services that previously ran in privileged (i.e., supervisor) mode such as device drivers, file systems, network stacks, etc., now run in non-privileged (i.e., user) mode. Because a separation kernel provides very specific functionality, the security policies that must be enforced at this level are relatively simple. The primary concerns of a separation kernel are the partitioning of processes and data plus the containment of systemwide failures. Consequently, we can capture the security requirements for a separation kernel by four foundational security policies:

- **Data Isolation.** Information in a partition is accessible only by that partition, and private data remains private.
- **Control of Information Flow.** Information flow from one partition to another is from an authenticated source to authenticated recipients; the source of information is authenticated to the recipient, and information goes only where intended.
- **Periods Processing.** The microprocessor and any networking equipment cannot be used as a covert channel to leak information to listening third parties.
- **Fault Isolation.** Damage is limited by preventing a failure in one partition from cascading to any other partition. Failures are detected, contained, and recovered locally.

The resultant kernel is now much smaller and simpler, and conducive to rigorous inspection and mathematical proof of correctness by techniques such as formal methods. This size reduction is an instantiation of MILS's most fundamental benefit: *Dramatically reduce the amount of security-critical code so that we can dramatically increase the level of rigor when we inspect that code.* If we are doing very few things, we should be able to do them very well, so well, in fact, that the code can be *trusted* to protect our most valuable data under the highest level of threat.

MILS middleware is an expansive concept with a very broad user-mode layer. It contains many operating system services such as device drivers that previously ran in privileged mode. Running in user mode, they are subject to the kernel's security policy enforcement. MILS middleware also includes functions traditionally thought of as being one level removed from the core operating system: file systems, network stacks, common libraries, encryption, authentication, etc. MILS middleware also includes traditional application-level middleware technologies such as Common Object Request Broker Architecture (CORBA) [8], Data Distribution Service (DDS), Web services, etc. MILS middleware resides in the same user-mode partition as the application that it supports or in protected user-mode partitions by itself.

Applications do their processing and *enforce their own security policies* in user-mode partitions. Applications running in their partitions can only access the memory that has been explicitly allocated for each partition. Application partitions can only communicate with each other through paths that have been configured when the system was generated. Under no circumstances may application partitions access hardware directly unless explicitly authorized to do so. The MILS architecture, along with a notional set of allowable information flows, is illustrated in Figure 1.

Why is application-level security-policy enforcement effective in MILS when it was not effective by itself in traditional monolithic architectures? It is because the MILS separation kernel guarantees control of information flow and data isolation. It makes this guarantee for the first time at an assurance level that was next to impossible to achieve with the monolithic kernels. Due to technology advances in both smaller circuits and increased functionality, we now have processors powerful enough to handle the context switching required for MILS, while still maintaining system performance.

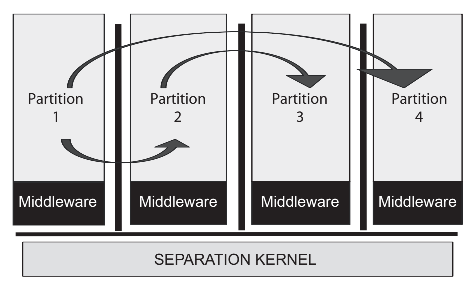
In the last 15 years, the number of context switches per unit of time that a state-of-the-art microprocessor can handle has increased by a factor of 1,000. Processor speed has increased by a factor of 100. The number of transistors per cubic inch on a wafer has increased by a factor of 125. We can now perform 50,000 context switches at a cost of 5 percent of the microprocessor clock. This 5 percent is the MILS security and safety tax.

Because information originates only from authorized sources, is delivered only to the intended recipients, and the source is authenticated to that recipient, the application developer is empowered to build his or her own reference monitors[2] at the application layer that include the following:

- **Non-bypassable.** Security functions cannot be circumvented.
- **Evaluatable.** Security functions are small and simple enough to enable rigorous proof of correctness through mathematical verification.
- **Always Invoked.** Security functions are invoked each and every time.
- **Tamperproof.** Security functions and their data cannot be modified without authorization, either by subversive or poorly written code.

An acronym for these four attributes is *NEAT* [9]. Security policy enforcement that is not NEAT is not effective. Although other operating systems have offered some form of non-bypassability and tamperproof functionality, MILS provides

Figure 1: *MILS Architecture Information Flows*



Common Terms

API	Application Programming Interface	MMU	Memory Management Unit
CIK	Crypto Ignition Key	MSLS	Multiple Single Levels of Security
CORBA	Common Object Request Broker Architecture	PCS	Partitioning Communications System
DDS	Data Distribution Service	SOAP	Simple Object Access Protocol
EAL	Evaluation Assurance Level	RTOS	Real-Time Operating System
HAL	Hardware Abstraction Layer	TCP/IP	Transport Control Protocol/ Internet Protocol
HTTP	Hyper-Text Transfer Protocol	UDDI	Universal Description, Discovery, and Integration
IPv6	Internet Protocol version 6	WSDL	Web Services Description Language
MILS	Multiple Independent Levels of Security	XML	Extensible Markup Language
MLS	Multiple Levels of Security		

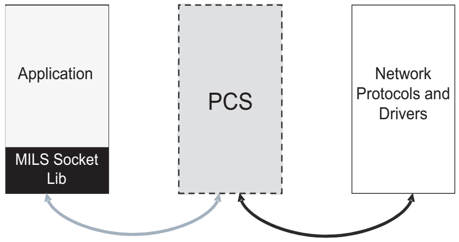


Figure 2: *Secure Network System Configuration*

*NEAT*ness for the first time in a COTS package that is formally modeled and mathematically verified at a high assurance level.

Divide and Conquer

The duration, schedule risk, and cost of evaluating, certifying, and deploying software increase non-linearly with the size of the code. These increases are especially onerous at high levels of assurance. Guaranteed NEATness enables us to design a MLS or Multiple Single Levels of Security (MSLS)[3] system as a set of independent system high partitions with cross-domain servers, downgraders, and guards enabling secure communications both among those partitions and also with external systems.

Another MILS objective is to enable the evaluation and certification of a complex system to be broken down into a number of independent, small evaluations. Security-critical software components that handle more than one level of information can be evaluated at high levels of robustness, approximately Evaluation Assurance Level (EAL) 6+ of the Common Criteria, an internationally approved set of security standards [10]. Cross-domain servers, downgraders, and guards, leveraging NEATness, can be small and tightly focused, making high-assurance evaluations of those components practical, achievable, and affordable. Single-level partitions, which each deal with only one level of information, can be evaluated at medium levels of robustness, approximately EAL 4, which is practical and achievable for large bodies of code. The independence of these evaluations also enables reuse of code, reuse of application programming interfaces (APIs), reuse of specifications, reuse of evaluation artifacts, and reuse of certifications to the greatest degree possible.

Connecting to Other Systems

MILS network components such as protocol stacks and their associated interface device drivers can be put into partitions of their own. This architecture has several advantages:

- Network facilities can be used by multiple application partitions.
- Network data is processed in unprivileged user mode, eliminating a vulnerability that is a common avenue of attack.
- Complex protocol code such as Internet Protocol (IP) Ver. 6 can be evaluated and certified independent of the applications using the code, enabling reuse of the evaluation artifacts.

Applications use an API to interact with the network. The MILS network API can have the same semantics as in a traditional operating system such as the familiar Transport Control Protocol/IP socket calls. The API implementation difference can be completely *under the hood*, transparent to the application developer. Instead of interacting directly with the protocol, a MILS socket implementation uses the separation kernel's interpartition communications facility to forward outgoing data to the protocol stack. Incoming data is handled similarly in the opposite direction.

Secure Network Systems

The network is the platform. The embedded computer that is not connected to another processor is a rare exception. By enforcing its four foundational security policies, MILS implements a robust information assurance foundation in a single node. We can then implement a robust information assurance foundation throughout a distributed system by providing *end-to-end* enforcement of those same security policies. End-to-end enforcement is provided by a high-assurance middleware component called the Partitioning Communications System (PCS). Leveraging the separation kernel's guarantee of controlled information flow within a single node, the PCS is always interposed between an application and the protocols/drivers that effect an off-board data transfer. The configuration is illustrated in Figure 2.

The PCS enforces the security policies end-to-end by providing the following:

- Strong identity of each node within a collection of MILS nodes (an enclave).
- Separation by level and/or community of interest. Enclaves are then connected together via high-assurance MILS cross-domain servers.
- Secure configuration, validating that all security databases are consistent.
- Secure image loading.
- Secure clock synchronization.
- Provisioning of bandwidth and quality of service.
- Suppression of covert channels.

Network Middleware Tools

While we are developing new systems that enable the warfighter to share information, it is important to not reinvent the wheel. Distributed system solution designers make frequent use of COTS network middleware for various application paradigms:

- Client/server, often using CORBA [11], distributes logic.
- Publish/subscribe, often using DDS, distributes data.
- Web-enabled services, using HyperText Transfer Protocol servers and components such as Extensible Markup Language; SOAP [Simple Object Access Protocol]; Web Services Description Language; Disco; Universal Description, Discovery and Integration; etc. that enable the communication between large diverse distributed communities of interest.

All of these technologies can be viewed as tools that provide the application programmer with a higher level abstraction to the rudimentary socket interface. Much of the code for these networking middleware technologies can either reside together in the same partition as the application that it supports, or it can reside in a partition by itself. In either case, porting existing code to a MILS environment is a straightforward task because the socket API does not need to change. The PCS still fits between the network middleware and the protocol stack and/or device drivers.

The system architect should not view the use of CORBA, DDS, or Web services as mutually exclusive. A single application can use CORBA for remote invocation, for distribution of logic, and for *smart pull* of needed information; it can use DDS for *smart push* of sensor data that is being monitored; and it can use Web services as a graphical interface for reports from one large community of interest to another, e.g., the Army infantry reporting threat data and the Air Force monitoring Web reports and providing air support.

There is an interesting side benefit to combining network middleware with the PCS. One of the purposes of network middleware is to make the number and location of processors sharing traffic as transparent as possible to the application. At the same time, in a secure networked system, we need to know exactly where our data came from and where it will go. Merging PCS functions with network middleware such as CORBA or DDS gives the system designer the flexibility to relocate system functions without introducing new threats to data confidentiality or integrity. This enables ad hoc networks and coalitions to be formed based on newly identified threat data, and to be dissolved as soon as the threat is dealt with.

Security is required when fielding systems that are either mission-critical or use national information. At the same time, there is a massive investment in applications using traditional operating systems and traditional middleware. The MILS architecture can provide a high-assurance foundation for fielded systems while preserving much of the legacy code base.

Guest Operating Systems

A traditional operating system, either an embedded real-time operating system (e.g., INTEGRITY, VxWorks, or LynxOS) or a desktop operating system (e.g., Linux, Windows, or Solaris) can run inside a MILS partition as a *guest operating system*. Operating systems written with portability in mind have a hardware abstraction layer (HAL) that localizes all processor-specific functions. Writing a new HAL is the major task in porting to a new central processing unit. You can use that same expertise to write a HAL that abstracts the MILS separation kernel as the hardware to the guest.

By itself, the guest operating system concept enables legacy applications to be easily ported to the high-assurance MILS environment. Another possibility is that multiple MILS partitions can each contain an instance of the guest operating system. This effectively creates multiple virtual operating systems on a single real microprocessor. The MILS separation kernel provides *trustworthy* separation with respect to both memory access and central processing unit time. Communications among the partitions is limited to those paths explicitly created when the system was generated. This is a practical path to implementation of cross domain solutions. It is also a practical path to implementation of high-assurance workstations suitable for MLS or coalition force operations.

For example, inside a partition the guest operating system can run as a thin client; it can be downloaded from a remote server. Which remote server a thin client is downloaded from can be determined from a token reader or crypto ignition key. The token would indicate nationality, clearance, and job title. The PCS would open a secure connection to a server that the user, who inserted and unlocked the token, was authorized to communicate with. Access to the local devices such as screen, keyboard, mouse, hard drive, etc., would be provided by the MILS workstation.

Supporting the Warfighter

The Air Force Research Laboratory (Information Directorate), in cooperation with the National Security Agency, Department of Defense prime contractors, academia, and software suppliers, is managing a MILS program to combine the best of existing commercial standards for flight safety and integrated modular avionics with the following:

- DO-178B, Software Considerations in Airborne Systems and Equipment Certification, Level A [12].
- ARINC-653, Avionics Application Software Standard Interface [13].

The program is also combined with the following appropriate standards for security:

- Common Criteria (International Organization for Standardization 15408), EAL 6, augmented [10].
- Director of Central Intelligence Directive 6/3, Protecting Sensitive Compartmented Information Within Information Systems, Protection Level 5 [14].

There is significant synergy among these standards. While they each have a specific area of interest, there is a great deal of common ground between safety-critical and security standards with respect to sound engineering practice, meeting requirements, and having the plans in place to address flaws.

The participating software suppliers that are currently developing MILS separation kernels are, alphabetically, Green Hills Software, Inc. [15], LynuxWorks, Inc. [16], and Wind River Systems, Inc. [17]. Objective Interface Systems, Inc. [18] is currently developing the partitioning communications system.

Putting It All Together

MILS is all about keeping things separate that need to be separate and doing so with components that we can trust with our most important data under the most severe threat. For security, we are keeping data separate by classification level, by community of interest, and by nationality. For safety, we are keeping applications separate by level of criticality. All of this is done with COTS software and certification artifacts that are reusable. Leveraging this reusability makes MSLS/MLS system development practical and certification/accreditation affordable and achievable. The end result is fielded systems that have high-assurance foundations but do not require custom-built security architectures for each new system.

For more information about MILS, please see <http://mils.ois.com>.◆

References

1. Vanfleet, Willard Mark, et al. "An Architecture for Deeply Embedded, Provable High Assurance Applications." May 2003.
2. Freeman, James, George Dinolt, and Richard Neely. An Internet System Security Policy and Formal Model. Proc. of 11th National Computer Security Conference, Oct. 1988: 10-19.
3. Anderson, Ross. Security Engineering. New York: Wiley & Sons, 2001.
4. Rushby, John. A Trusted Computing Base for Embedded Systems. Proc. of the 7th Department of Defense/NBS Computer Security Conference, Sept. 1984: 294-311.
5. Department of Defense. Trusted Computer System Evaluation Criteria (The Orange Book). DoD 5200.28-STD. Washington: DoD, 1983.
6. Moore, G. "Cramming More Components Onto Integrated Circuits." Electronics Magazine 19 Apr. 1965.
7. Rushby, John. "The Design and Verification of Secure Systems." ACM Operating Systems Review 15.5.
8. Rushby, John. "Proof of Separability: A Verification Technique for a Class of Security Kernels." Computer Science 137: (1982) 352-367.
9. Partitioning Kernel Protection Profile, May 2003.
10. Common Criteria for Information Technology Security Evaluation, Ver. 2.1, 19 Sept. 2000.
11. Currey, Jonathan, Bill Beckwith, et al. Real-Time CORBA 1.1 Specification, Object Management Group Aug. 2002.
12. RTCA <www.rtca.org>.
13. ARINC <www.arinc.com/cf/store/index.cfm>.
14. Director of Central Intelligence. "Protecting Sensitive Compartmented Information Within Information Systems." Directive 6/3. Washington: DCID, 5 June 1999 <www.fas.org/irp/offdocs/DCID_6-3_20Policy.htm>.
15. Green Hills Software <www.ghs.com>.
16. Lynux Works <www.lynuxworks.com>.
17. Wind River Systems <www.windriver.com>.
18. Objective Interface <www.ois.com>.

Notes

1. Orange book levels began with A1 and moved down through levels B3, B2, B1, C2, and C1.
2. A reference monitor is an Access Control concept referring to an abstract machine that mediates all accesses to objects by subjects [3].
3. MSLS means there are multiple channels each with their own separate data classification [9].

About the Authors

W. Mark Vanfleet has worked for the National Security Agency (NSA) Information Assurance Directorate for 18 years as an information systems security analyst and mathematician. He holds NSA certifications in crypto-mathematics, communication and information systems security, and software engineering process and practice. Vanfleet has been involved in high-assurance software architecture, design, and evaluation for 25 years. He has bachelor's degrees in mathematics and computer science, and a master's degree in mathematics and statistics from the University of Utah.

National Security Agency
9800 Savage RD STE 6709
Fort Meade, MD 20755-6709
Phone: (410) 854-6361
E-mail: wvanflee@restarea.ncsc.mil

Jahn A. Luke is a senior program manager at the Embedded Information Systems Branch, Information Directorate, Air Force Research Laboratory (AFRL) where he is lead for legacy system modernization and manager of the Multiple Independent Levels of Security (MILS) program. He has more than 29 years of hardware and software experience at AFRL in the development of technologies for real-time simulation systems and embedded software systems. In addition to MILS, he currently manages projects addressing the upgrade of legacy embedded computer systems for such programs as F/A-22, CV-22, and F-117. Luke has a Bachelor of Electrical Engineering from the University of Detroit.

AFRL/IFTA
2241 Avionics CIR BLDG 620
Wright Patterson, OH 45433
Phone: (937) 255-6653 ext. 3585
Fax: (937) 656-4277
E-mail: jahn.luke@wpafb.af.mil

R. William Beckwith is the chief executive officer and chief technology officer of Objective Interface Systems, Inc. He has been engineering software for over two decades with the last decade focused on embedded and real-time systems. Beckwith is a frequent speaker on real-time, embedded, and high-assurance software in North America, Japan, and Europe. He is currently involved in developing software and standards for high-assurance embedded systems. Beckwith continues to lead the initiative for the advancement of Common Object Request Broker Architecture and Multiple Independent Levels of Security in the real-time and embedded communities.

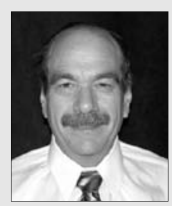
Objective Interface Systems
13873 Park Center RD STE 360
Herndon, VA 20171-3247
Phone: (703) 295-6519
Fax: (703) 295-6501
E-mail: bill.beckwith@ois.com

Carol Taylor, Ph.D. is a professor of computer science at the University of Idaho where she currently teaches classes in computer security and does research. Taylor's research interests are in computer security and software engineering with a special emphasis on high-assurance systems. Her research background includes projects in survivability, intrusion detection, formal methods, and multiple levels of security policy. Prior to obtaining her doctorate, Taylor held programming/analyst positions.

University of Idaho
Computer Science Department
Moscow, ID 83844
Phone: (208) 885-5276
Fax: (208) 885-9052
E-mail: ctaylor@cs.uidaho.edu

Ben Calloni, Ph.D., is a senior software and avionics researcher for Lockheed Martin (LM) Aeronautics Company in Fort Worth, Texas. He has been addressing the feasibility of using standards-based commercial software in mission-critical avionics systems since joining LM in 1997. The past three years he has been investigating the use of commercial off-the-shelf-based security solutions (Multiple Independent Levels of Security program) for the various LM Aero weapon systems. Calloni serves on the board of directors for and is active in both the Object Management Group and The Open Group: International Standards Consortia. He has degrees in industrial engineering and computer science from Purdue University and a doctorate in computer science from Texas Tech University.

Lockheed Martin Aeronautics Co.
P.O. Box 748
MZ 8604
Fort Worth, TX 76101
Phone: (817) 935-4482
Fax: (817) 762-6784
E-mail: ben.a.calloni@lmco.com



Gordon Uchenick is a frequent presenter and lecturer on Multiple Independent Levels of Security (MILS) as well as an author on the subject. Uchenick also participates in the MILS community standards bodies such as The Open Group. Prior to joining Objective Interface, he was an engineering specialist with Wind River Systems, concentrating on the company's security technologies.

Objective Interface Systems
13873 Park Center RD STE 360
Hendon, VA 20171-3247
Phone: (410) 256-7102
Cell: (410) 952-2739
Fax: (410) 256-7104
E-mail: gordon.uchenick@ois.com

Six Steps to a Successful COTS Implementation

Arlene F. Minkiewicz
PRICE Systems

A successful implementation of a commercial off-the-shelf-intensive software system can save programs money if you have the right solution and understand the potential risks involved.

Federal organizations are relying more and more on commercial applications to supplement, enhance, or replace proprietary systems. This dependency is driven by the promise of improved functionality and reduced total ownership cost, as well as concern over the lack of capability to develop and maintain proprietary information technology applications. However, failure to successfully select, control, and implement these critical components continues to result in projects that are delivered late and over-budget or that fail entirely.

The following *six-step methodology* highlights the important activities that should take place during a commercial off-the-shelf (COTS) implementation. Following this methodology throughout the software development life cycle will ensure that significant activities are not being ignored and will increase the chances of planning, executing, and deploying a successful COTS-based software solution.

One of the biggest problems sighted in COTS-based projects is a disconnect between time and cost expectations during planning and those actually realized.

During the planning stages, it is important to plan appropriately for all the major activities necessary to devise a well thought-out solution that will not fall apart with the first upgrade of one of its components. Research [1, 2, 3, 4, 5] has indicated the essential activities that must take place to ensure successful COTS-based projects:

- Analyze software requirements.
- Evaluate and select COTS solution(s).
- Negotiate terms with COTS vendor.
- Implement the COTS-based solution.
- Maintain and upgrade the COTS-based solution.
- Maintain license, subscription, and royalty fees.

Figure 1 portrays an overview of the six steps outlined above and highlights the interactions that may occur throughout the execution of these steps. While this diagram implies a time dependency between these steps, it is important to realize that in certain cases this is neither strictly adhered to nor are all the steps necessarily performed by the organization(s) contracted to deliver a solution. Some requirements analysis and COTS evaluation are likely to occur in very early stages of a project as feasibility and affordability are analyzed.

The following sections provide more details about each of these steps, along with a brief description of the factors to consider when evaluating the affordability and timeliness of a COTS-based solution. Specifics about the quantification and application of these factors can be found in [6].

1. Analyze Software Requirements

Software requirements analysis is a critical part of the software development process, although too often this activity is overlooked or glossed over in the rush to start *building software*. The requirements analysis process is necessary to determine what functionality is necessary to deliver the capability required by the eventual end-user(s).

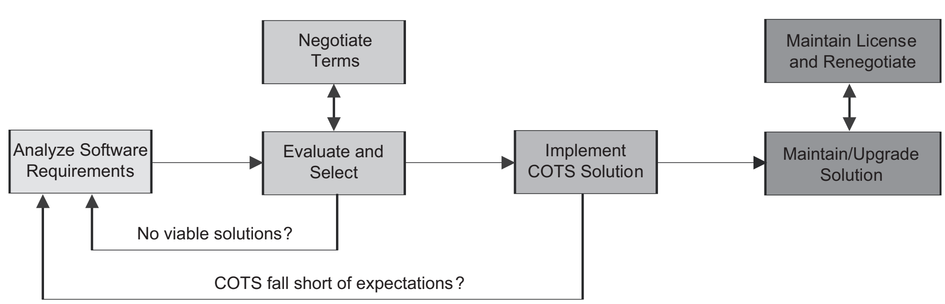
There are two general areas that need to be explored when determining and documenting requirements for a software system: end user requirements and technical requirements. The discovery process for end-user requirements involves business analysts or requirements engineers asking the end-user what they expect from the software. Once end-user requirements have been gathered, an important next step is for the business analysts or requirements engineers to restate those requirements and present them back to the end-user to ensure proper understanding. Technical requirements can be gathered through discussions with engineers who understand the technical nature of the problem being solved.

The question of whether or not COTS solutions are a viable alternative becomes an important factor during the software requirements analysis activity because the software requirements drive the selection criteria for potential COTS solutions. This being said, it is also important not to let the availability of COTS solutions cloud the analysis by obscuring requirements.

Solution providers should be aware that it is unlikely that any COTS solution will be available to satisfy all the requirements for a software system. The requirements analysis process should identify which requirements are the *must have* requirements and which can be somewhat *bendable*. During the evaluation, and possibly during implementation, tradeoffs will be necessary to compensate for functionality not available in COTS solutions (or promised and not delivered with a chosen COTS solution). Decisions should be made during the requirements analysis activity to determine which functions can be subject to such tradeoffs and which cannot. If most or all of the software requirements are determined to be *must haves*, it is wise to revisit the decision to pursue a COTS-intensive solution.

Although the focus of this article is on COTS software solutions, it is important to mention here that when entire systems are being constructed, the COTS decision may need to be visited even before soft-

Figure 1: *Overview of the Six Steps*



ware requirements analysis commences. During the analysis of system requirements, decisions may be required to determine whether certain functionality should be addressed with hardware or software. The availability of software COTS solutions could be a factor in the determination of the affordability of such tradeoffs.

As with all activities in a software development process, successful execution of the requirements analysis process takes time and effort. The major driver in determining time, cost, or effort for the software requirements analysis activity is a measure of the amount of functionality the software system is intended to deliver. The *measurement* of software size is always a challenge. A measure that quantifies functionality delivered such as function points or analogies to known systems is best.

Whether the plan is for that functionality to be delivered primarily with COTS solutions, newly developed solutions, or some combination of the two, the time and effort devoted to requirements analysis should be fairly consistent. The technical complexity of the functionality as well as the software's operational platform will also drive the requirements analysis effort because more complex solutions require more time to understand and communicate. Additionally, the presence of project constraints, timing, memory, or schedule will impact the effort required for this activity.

2. Evaluate and Select COTS Solution(s)

Once a decision to pursue a COTS alternative is made, the first step is to determine the availability of COTS solutions that have the potential to provide needed functionality, then evaluate these solutions. The main reasons to evaluate are the following:

- Determine whether the functionality promised is the functionality delivered.
- Determine whether system non-functional requirements (portability, reliability, security, performance) can be met.
- Determine whether functional requirements can be met by the functionality delivered.
- Determine whether a proposed suite of components can operate successfully in the environment(s) where the system is intended to operate.
- Determine whether a proposed set of components can operate successfully in an integrated fashion.
- Determine the stability and viability of the vendor.
- Determine the willingness of the vendor to cooperate and help make the project successful.

The evaluation needs to be focused on more than just product characteristics such as functionality, maturity, technology, architecture, and long-term viability. There should also be a focus on vendor characteristics such as maturity, stability, cooperation, and ability to provide adequate support, training, and documentation. The evaluation should also be used to ensure that there are no compatibility issues associated with using COTS solutions from multiple vendors.

The selection process is often a combination of the following three evaluation techniques:

- **Progressive filtering of available COTS components.** This requires several iterations of filtering, each one going into progressively more detail until a single solution or set of solutions emerges as the best answer.
- **Puzzle assembly process.** This approach suggests that it is better to evaluate a set of components at one time using an evolutionary prototyping approach. In this method, multiple sets may be evaluated in parallel to identify which of them comes closest to solving the entire *puzzle* presented by the system requirements.
- **Identifying the keystone COTS software components.** This is identifying those components for which requirements (whether they be technical, process, functional, vendor, etc.) are unbendable, and then basing other component selections on compatibility and ease of interface with the keystone components.

"When performing a COTS evaluation, it is valuable to obtain references from the COTS vendors in an effort to speak with developers and end users who have worked with a particular COTS solution."

A commonly cited challenge by system integrators is that COTS products often fail to deliver the functionality or other requirements promised during evaluations. It is prudent to get some hands-on time with the components being evaluated where possible; this may be problematic as it most likely requires a great deal of cooperation and support from both the vendor and the integrating organization.

When performing a COTS evaluation, it is valuable to obtain references from the COTS vendors in an effort to speak with developers and end users who have worked with a particular COTS solution. This not only provides valuable feedback about vendors' responsiveness and dependability, it also aids the planning process by highlighting problems or pitfalls other integrators may have experienced.

The evaluation and selection activity not only facilitates identification of available COTS solutions, it also points to those pieces of functionality that cannot be satisfactorily implemented by existing off-the-shelf solutions. An important byproduct of this investigation may be an examination of the cost, schedule, and effort associated with developing custom code to make up for required functionality missing in COTS solutions. This examination may require revisiting the cost/benefit analysis leading to the decision to build a COTS-based solution.

Generally, the time and effort devoted to the selection and evaluation activity is often based on a predetermined level of effort. The determination of this level of effort should consider the amount of functionality being implemented with COTS solutions (based on functional size or analogy), the number of solutions that will be evaluated, the type(s) of evaluations being performed, and the number and criticality of evaluation criteria.

3. Negotiate Terms With COTS Vendors

Certainly it is important to negotiate the best deal possible when working with one or more vendors to craft a solution. It is even more important to understand the impact of these negotiations and their timing on the eventual success or failure of your project. Several of the most commonly cited challenges of those building software solutions with COTS components involve vendor forthrightness and cooperation [4].

Vendors are much more likely to address customer concerns with missing or incomplete functionality and/or bugs in the software before signing on the dotted line. During the negotiation process, it is important to address and resolve any known issues and establish expectations for issues that emerge during the integration process and throughout the product life cycle. Clearly, the size of the vendor and the size of the purchase may be factors in determining how demanding any particular customer can be, but it is important to set expectations with all of the project stakeholders.

The end result of this negotiation should be a clear picture of the non-recurring and recurring costs associated with the system being developed. A nonrecurring cost is a one-time fee associated with acquiring the COTS solution such as the purchase price of shrink-wrapped software. Recurring costs are those generally based on usage or time of use such as annual licensing fees. Negotiations should also result in a common understanding between parties of update and upgrade policies, as well as expectations with regard to vendor responsiveness and cooperation. It may also be necessary during this step to develop a plan with the vendor to ensure that maintenance of the deployed solution be possible even if the vendor goes out of business. This plan generally involves the escrowing of source code to be made available only under terms of the agreement such as bankruptcy or company demise.

The cost and effort drivers for this activity should be broken into two parts. The first part is the actual dollar value that is determined for delivery of the COTS component after negotiations and any promised royalties and other fees. The second part relates to the amount of time and resources that must be devoted to the negotiation with the vendor.

4. Implement the COTS-Based Solution

Once analysis, evaluation, and selection of a COTS-based solution are complete, implementation can commence. The following activities may be required to ensure successful implementation.

Tailoring of a COTS Solution

There are certain necessities that should be performed in or around software to get the COTS software components configured for the system and its requirements. Databases and other parameters need to be initialized and loaded, all or part of the components need to be registered with the operating system, security must be activated or initialized, screens and reports need to be scripted, and other script development may be required.

Although these activities are unlike traditional coding exercises, they do take time and effort to complete. The results of these activities require testing and verification. These tasks also require a significant level of understanding as to how the COTS component(s) work and how to work with them. This requires reading manuals and/or attending training and then experimenting with the actual components to reach a level of competency.

The major cost, effort, and schedule drivers for the tailoring activity include the amount and complexity of scripts, database parameters, reports, and screens being customized. Additionally, as security and access control requirements increase in rigor, they will drive up time and cost. It is important also to consider the ease with which the COTS solution can be understood, the quality of training and documentation, and the integration team familiarity with the COTS solutions being used and the system being implemented.

Modification of COTS Software

Generally the definition of COTS software precludes modifications because COTS software does not have source code available. This is the case when the COTS software is a shrink-wrapped commercial product. Sometimes solutions call for the integration of a series of off-the-shelf components that do not meet this traditional definition of COTS but rather are components with source code available that are either furnished by the customer or otherwise obtained. While these projects are not strictly *COTS* projects, they do happen and are mentioned here for completeness.

It would be nice if the COTS software completely satisfied all the functional requirements it was selected to meet, but this is often not the case. In situations where the source code can be made available, the project team needs to make a determination whether or not modification is an option. It is generally a bad idea to make modifications to COTS software because this negates much of the productivity increase obtained from using COTS components and is likely to jeopardize any likelihood of supplier maintenance of the COTS components. If COTS modifications are being considered, care should be taken to ensure that these modifications are very modular in nature. The developers need to learn a great deal about the architecture and basic structure of the solution before any modifications can be made. It is important to understand that the project productivity hit can be substantial even when the slightest modifications are made to a COTS component.

The major cost, effort, and schedule drivers for modifications to COTS software are the same factors that drive costs for any software modification project (functional size, extent of modification, technical complexity, eventual operating platform, productivity, and efficiency of development organization). These factors must then be augmented to account for unfamiliarity of the COTS solution code, quality of COTS training and documentation, vendor cooperation, development team experience with COTS solutions, and the COTS integration process. It is also important to include maintenance of the entire COTS component in the affordability analysis as once modifications have been made it is unlikely that the COTS vendor will continue to support their solution.

Design, Code, and Test of Glue Code

Glue code literally holds the system together. Glue code is any code that needs to be written to make the COTS software components function as advertised and/or as required. It is the code that references the interfaces in the COTS software component and needs to interpret return codes from these interfaces. Glue code is often required to convert data and other information from the format in which the system maintains data to the format required by the COTS component. In a well-written application, the glue code acts as a layer between the system and COTS components, encapsulating the data in such a way that upgrades and replacements are as painless as possible. Finally, glue code is sometimes required to add functionality that the COTS software component implements inadequately or that should be provided by the COTS component but is not.

These development activities are complicated due, primarily, to unavailable source code. The complexity of glue code development is akin to a situation where an entirely new team of developers is brought in to integrate custom built components, but is given limited or no access to the original development team and the source code. The unfamiliarity of the interfaces, along with the inability to debug the components, adds complexity to the development exercise.

Another factor that makes glue code development differ from traditional code development is the complete reliance on vendors to fix bugs when they are discovered, ensure that upgrades are upwardly compatible, release stable products, and fill in where documentation falls short. Bug fixing can be particularly problematic in a COTS project, especially when there are multiple vendors involved or when modifications have been made to the COTS components. With multiple vendors, especially with unavailable source code, the integrator must rely on the vendors not only to fix the bugs but also to cooperate with each other in the identification of where a bug actually resides. When the COTS components have been modified, the integrator often must struggle to convince the vendor that the bug is in the original code and not a side effect of changes they have made. Careful modularization and documentation of any COTS modifications may help alleviate this problem.

As with modifications to COTS components, the major cost, effort, and schedule drivers for glue code development are not unlike those drivers associated with any custom development, but the productivity of the development team needs to be adjusted to account for unfamiliarity and unavailability of COTS components along with the requirement for vendor support and cooperation in solving integration problems. As the number of different COTS components and the number of different vendors involved increases, so increases the complexity (thus effort and schedule) of this activity.

Integration and Test of COTS Components With Other COTS or Custom Components

The system needs to be integrated and tested to ensure that all functional and non-functional requirements are met. This activity tests the entire system (or subsystem) as a complete unit, verifying that there are no system-level problems associated with incompatibilities or competing demands of components for limited resources.

Requirements related to performance, reliability, and security could be particularly problematic during this activity as these types of problems are likely to go unnoticed until the whole system is running together. It is best to use an incremental or evolutionary approach when building COTS-inclusive systems, as these approaches advocate successive integrations during development rather than a waterfall-type process where integration does not happen until the end of the development.

A well-designed integration and test approach would focus early releases on those areas that are high risk or most likely to lead to incompatibilities. If multiple COTS components from multiple vendors are being integrated, it is important to have all of them running together at the same time as early as possible, even if each is functioning in very limited capacity with respect to its intended feature set. This helps identify potential contention and incompatibilities between components.

When assessing the effort and schedule for glue code development, tailoring, or integration activities, an important factor is the update cycle for the COTS solutions being integrated. If updates/upgrades are frequent, additional time and effort may be required to evaluate and possibly incorporate these updates.

> *"It is generally a bad idea to make modifications to COTS software because this negates much of the productivity increase obtained from using COTS components and is likely to jeopardize ... supplier maintenance ..."*

Upgrades and updates may be included if they contain required bug fixes, improvements to keep the look and feel current with user expectations, or features that relate to missing or incomplete requirements in earlier versions. Vendor quality and stability, along with vendor(s) regular release schedules, are important factors in assessing effort associated with updates and upgrades.

In general, the effort, cost, and schedule for many system-level integration activities is likely to be *higher* for COTS-inclusive software than software that is composed of custom-built components because of the unfamiliarity with the code. The COTS software component should be viewed by the integrator as a *black box*. The factors that drive the effort and schedule for integration activities include the amount of functionality being integrated (including functionality provided by homegrown development as well as that coming from the COTS components), the complexity of the functionality, the operating platform for the system, glue code size, and vendor-related issues such as cooperation, support, and upgrade policies.

5. Maintain and Upgrade the COTS-Based Solution

Once the software is deployed, the following two ongoing activities are required to keep it operational and keep end-users happy.

Evaluation and Inclusion of Updates and Upgrades From the Vendor

There are countless reasons why the inclusion of updates and upgrades are desirable once the product is deployed. If there are bugs in the COTS software that affect system operation, then an upgrade is obviously needed. Beyond this, the vendor should be upgrading the product to keep up with rapidly changing technology. To maintain a software system that meets market expectations with respect to performance, look and feel, operating platforms, etc., updates of the COTS software components will be likely.

Whether including an update or upgrade, each refresh of the COTS components poses potential risk. Assessment is required with each release to determine whether it makes sense to include it in the software system. Sometimes upgrades change the interfaces to the COTS software components so that with the upgrade, existing functionality ceases to work correctly, requiring a rewrite of some of the glue code. Vendors find that in order to offer new features or to keep up with technology, they must change interfaces and databases or rework functionality. To get access to the new features and technology in an upgrade, the software developer may be forced to accept changes he or she does not want or need as well – creating additional cost with no added value to the software system.

Every upgrade also carries with it the possibility of incompatibilities with the existing software system, other COTS software components, or even the operating platforms on which the system runs. For this reason, each upgrade should include a repeat of system operational testing and system regression testing to ensure that the effects on system operation are understood and desirable. While it is important to understand all of this when evaluating whether or not to upgrade, it is also important to understand that there is a risk associated with failure to upgrade. At some point, the vendor will discontinue support of older versions of the COTS software.

The major factors that drive cost, effort, and schedule for the evaluation of COTS upgrades include the amount of functionality delivered by COTS solutions, the number of COTS solutions in the system, the upgrade/update frequency of the vendor(s), and the quality and stability of the COTS solutions. When (and if) a decision has been made to include upgrade/updates of COTS solution(s), the drivers for the integration and test are the same as those for integration and test during development, although the extent of effort should be tempered by the extent of functionality changes in the upgrades/updates.

Bug Fixes

Software, whether it is homegrown or acquired externally, is likely to have defects. Bug fixing efforts for COTS-intensive systems will differ significantly from typical repair efforts. Additionally, bugs may exist in the COTS software component that the vendor is unable or unwilling to fix for which workarounds in the glue code need to be developed.

The effort associated with bug fixes for COTS software, as with any software, is a direct function of the quality of the initial software offering, as well as the quality with which bugs are fixed. This quality is generally a function of the amount of functionality, the complexity of the system, and the development processes and practices employed by the initial software developer. For COTS software, some of these factors are apparent and some are hard – if not impossible – to find out. Also, how and when the bugs in the COTS component get fixed is for the most part out of the integrators' control.

These factors complicate the process of planning for maintenance of COTS-based systems. Additionally, the maintenance process is plagued with the same issues cited earlier for the glue code design, integration, and test: It is not always obvious where the bugs actually occur. Despite the hurdles mentioned, it is still possible to plan proactively for the maintenance of a COTS-based system based on what is known. Functional size of the entire system (including not just COTS components but homegrown components as well), system complexity, operating platform, glue code size, amount of modification, and maintenance team productivity can be used to baseline the effort. This effort should then be adjusted for the loss of productivity associated with debugging through *black box* code and interfacing with multiple vendors.

6. Maintain License, Subscription, and Royalty Fees

License or maintenance fees need to be paid in order to ensure updates and upgrades as well as continuing support of the COTS components. It is important to understand vendor(s) upgrade policies. It is also wise to do a long-term analysis of the differences between annual subscription fees (if subscription is an option) versus paying for upgrades on an individual basis. This analysis should include upgrade policies, vendor stability, and frequency of releases. License and royalties should be an important part of the initial negotiation process. Renewal periods are an opportunity to revisit the terms of the negotiation to determine whether the vendor is meeting support and upgrade commitments.

Conclusions

A well thought-out and well-executed software project that incorporates one or many COTS solutions can happen more quickly and be more cost effective than the same system implemented with custom developed components. Too often, COTS projects are not thought out or planned, running on the incorrect assumption that every COTS solution is a small integration project without the issues and complexities cited above. This way of thinking leads to unrealistic and poorly managed expectations, resulting in failed projects. These types of failures occur when projects fail to plan for or incorporate the additional activities unique to COTS-intensive developments. Following this *six-step methodology* will ensure that important activities and decision points are properly executed, reducing many of the risks associated with such developments.◆

References

1. Ellis, T. "COTS Integration in Software Solutions – A Cost Model." INCOSE Symposium, St. Louis, MO, July 1995.
2. Center for Software Engineering. "COCOTS." Los Angeles, CA: University of Southern California, June 1997 <http://sunset.usc.edu/research/COCOTS/cocots_main.html>.
3. Abts, Christopher. "COTS Software Integration Cost Modeling Study." Los Angeles, CA: Center for Software Engineering, June 1997.
4. Brownsword, L., et al. "Lessons Learned Applying Commercial Off-the-Shelf Products." Pittsburgh, PA: Software Engineering Institute, June 2000 <www.sei.cmu.edu>.
5. Oberndorf, P., et al. "Workshop on COTS-Based Systems." Software Engineering Institute, Nov. 1997 <www.sei.cmu.edu>.
6. Minkiewicz, A. The Real Costs of Developing a COTS-Based System. Proc. of IEEE Conference on Aerospace and Defense, Big Sky, MT., Mar. 2004.

About the Author

Arlene F. Minkiewicz is chief scientist of the Cost Research Department at PRICE Systems. She is responsible for the research and analysis necessary to keep the suite of PRICE estimating products responsive to current cost trends. In her 20-year tenure with PRICE, Minkiewicz has researched and developed the software cost estimating relationships that were the cornerstone for PRICE's commercial software cost estimating model, ForeSight, and invented the Cost Estimating Wizards originally used in ForeSight that walk the user through a series of high-level questions to produce a quick cost analysis. As part of this effort she has invented a sizing measurement paradigm for object-oriented analysis and design that allows estimators a more efficient and effective way to estimate software size. She recently received awards from the International Society of Parametric Analysts and the Society of Cost Estimating and Analysis for her white paper "The Real Cost of COTS." Minkiewicz contributed to a new parametric cost estimating book with the Consortium for Advanced Manufacturing – International called "The Closed Loop: Implementing Activity-Based Planning and Budgeting," and she frequently publishes articles on software estimation and measurement. She has also been a contributing author for several books on software measurement and speaks frequently on this topic at numerous conferences.

17000 Commerce PKWY STE A
Mt. Laurel, NJ 08054
Phone: (856) 608-7222
Fax: (856) 608-7247
E-mail: arlene.minkiewicz@pricesystems.com

Performance-Based Earned Value

Paul J. Solomon
Northrop Grumman Integrated Systems

Performance-Based Earned Value® (PBEV℠) is an enhancement to the Earned Value Management Systems (EVMS) standard [1]*). PBEV overcomes the standard's shortcomings with regard to measuring technical performance and quality because it is based on standards and models for systems engineering, software engineering, and project management. The distinguishing feature of PBEV is its focus on the customer requirements. PBEV provides principles and guidance for cost-effective processes that specify the most effective measures of cost, schedule, and product quality performance.*

Program managers (PMs) expect accurate reporting of integrated cost, schedule, and technical performance when the supplier's Earned Value Management Systems (EVMS) complies with the EVMS guidelines in the American National Standards Institute (ANSI)/ Electronic Industries Alliance (EIA) Standard-748-A-1998. However, EVM data will be reliable and accurate only if the following occurs:

- The indicated quality of the evolving product is measured.
- The right base measures of technical performance are selected.
- Progress is objectively assessed.

Using EVM also incurs significant costs. However, if you are measuring the wrong things or not measuring the right way, than EVM may be more costly to administer and may provide less management value [2].

EVMS Shortcomings

The EVMS standard has significant shortcomings with regard to standards and models for systems engineering (SE), software engineering, and project management. Consequently, there is no assurance the reported earned value (EV) is based on product metrics and on the evolving product quality as defined by the standards and models.

First, the EVMS standard states that EV is a measurement of the quantity of work accomplished and that the quality and technical content of work performed are controlled by other processes. A PM should ensure that EV is also a measurement of the product quality and technical maturity of the evolving work products instead of just the quantity of work accomplished. However, a Naval Air Systems Command (NAVAIR) organization that used EVM and the Team Software Process℠ to accelerate software process improvement concluded that EVM did not address product quality and was not beneficial at the higher levels of the Capability Maturity Model® (CMM®) for Software [3].

Second, the EVMS principles address only the project work scope. EVMS ignores the product scope and product requirements.

Third, EVM is perceived to be a risk management tool. However, EVMS was not designed to manage risk and does not even mention the subject.

The following guidance will enable a PM to use Performance-Based Earned Value® (PBEV℠) to overcome the limitations of EVMS and provide a framework for utilizing PBEV as a key component of project planning, measurement, and control. The guidance is based on actual project experience and has contributed to the success of software-intensive programs, including the B-2 stealth bomber.

Table 1: *Department of Defense System Engineering Policy and Guides*

DoD SE Policy and Guides	Policy	DAG	SEP Guide
Develop systems engineering plan.	P	4.2.3.2	1.0
Event-driven timing of technical reviews.	P	4.5.1	3.4.4
Success criteria of technical reviews.	P	4.5.1	3.4.4
Assess technical maturity in technical reviews.		4.5.1	3.4.4
Integrate SEP with integrated master plan.		4.5.1	3.4.5
Integrate SEP with integrated master schedule.		4.5.1	3.4.5
Integrate SEP with technical performance measures (TPM).		4.5.1	3.4.4
Integrate SEP with earned value management.		4.5.1	3.4.5
Use TPMs to compare actual versus planned technical development and design maturity.		4.5.5	3.4.4
Use TPMs to report degree to which system requirements are met in terms of performance, cost, and schedule.		4.5.5	3.4.4
Use standards and models to apply systems engineering.		4.2.2, 4.2.2.1	
Institute requirements management and traceability.		4.2.3.4	3.4.4
Use EVM.		11.3.1	

Department of Defense Policy

Compliance with SE standards will support the Department of Defense (DoD) acquisition policy that programs will implement SE plans (policy) [4]. The DoD also published the Defense Acquisition Guidebook (DAG) and the Systems Engineering Plan (SEP) Preparation Guide (SEP Guide) to provide discretionary best business practices to complement the policy. The SEP Guide cites engineering standards[1] that are sources of PBEV [5]. Table 1 shows pertinent policy components and implementing guidelines.

Product Metrics and Quality

The Institute of Electrical and Electronics Engineers (IEEE) 1220 and the EIA 632 have similar guidance regarding product metrics and quality. Product metrics allow assessment of the product's ability to satisfy requirements and to evaluate the evolving product quality against planned or expected values. Establishing a time-phased product quality requirements baseline against which progress can be measured normally precedes the schedule and budget. An exception for the system definition stage of the systems development life cycle, before the real product quality requirements are known, is discussed later. Of equal importance are a disciplined requirements traceability process and a requirements traceability database [6].

® Performance-Based Earned Value is registered with the U.S. Patent and Trademark Office by Paul Solomon.
℠ PBEV is a service mark of Paul Solomon.
® Capability Maturity Model and CMMI are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.
℠ CMM Integration and Team Software Process are service marks of Carnegie Mellon University.

Success Criteria

The standards discuss the importance of holding technical reviews at various stages of development to assure that all success criteria have been met. IEEE 1220 provides success criteria to be used at major technical reviews. For example, some of the success criteria for a preliminary design review are the following:

- Prior completion of subsystem reviews.
- Determine whether total system approach to detailed design satisfies the system baseline.
- Unacceptable risks are mitigated.
- Issues for all subsystems, products, and life-cycle processes are resolved.

The success criteria should be defined in a SEP or other technical plan. The customer should review this plan with the supplier and reach agreement on the success criteria to be used at technical reviews.

Technical Performance Measurement

Technical performance measurements (TPMs) are defined and evaluated to assess how well a system is achieving its performance requirements. TPM uses actual or predicted values from engineering measurements, tests, experiments, or prototypes. IEEE 1220, EIA 632 and "A Guide to the Project Management Body of Knowledge" (PMBOK Guide) [7] provide similar guidance for TPM planning and measurement and for integrating TPM with EVM. For example, EIA 632 states that TPMs predict the future value of key technical parameters of the end-system based on current assessments and that milestones are established for comparing planned and actual progress.

SE Work Products

The SE process generates significant work products that should be included in integrated planning and measured with EV. The process products of IEEE 1220 are as follows:

- Requirements baseline.
- Validated requirements baseline.
- Functional architecture.
- Verified functional architecture.
- Physical architecture.
- Verified physical architecture.

These, or similar, work products should be included in the integrated master schedule, be the output of work packages, and have defined success criteria.

CMM Integration

The CMM Integration℠ (CMMI®) [8] provides many practices that augment the EVMS guidelines. CMMI also lists typical work products (TWPs) within process areas. To ensure traceability of product quality requirements to work tasks and work products, these TWPs, or similar artifacts, should be the outcome of work packages. Here are some TWPs in CMMI.

Requirements development TWPs include the following:

- Derived requirements.
- Product requirements.
- Product-component requirements.
- Interface requirements.
- Activities diagrams and use cases.
- Results of requirements validation.

Technical solution TWPs include the following:

- Documented relationships between requirements and product components.

> *"The distinguishing feature of PBEV is its focus on the customer requirements ... Progress is measured against a plan to fulfill all customer requirements ... management is able to take rapid corrective actions on deviations that threaten customer satisfaction ..."*

- Product-component designs.
- Technical data packages.
- Allocated requirements.
- Verification criteria used to ensure requirements have been achieved.
- Interface control documents.
- Implemented design.

Verification TWPs include these:

- Exit and entry criteria for work products.
- Verification results.

A decision analysis and resolution TWP includes the results of evaluating alternate solutions.

Cost Savings

Measurement costs money. An enterprise must incur significant implementation and sustainment costs to use EVM. These costs can be reduced if the enterprise utilizes an effective process to determine what needs to be measured and limits the measurements to those that meet its information needs and objectives. Furthermore, management can be more effective if it focuses on fewer but more critical measures.

PBEV is cost-effective because it limits the number of activities that should be discretely measured to those that meet defined information needs such as the work products described above. Other measurable activities may be planned as level of effort, if it is not practicable to measure them, or they may be apportioned effort. Additional measurement guidance is available in a Software Engineering Institute technical note [9].

PBEV Characteristics

PBEV is a set of principles and guidelines that specify the most effective measures of cost, schedule, and product quality performance. It has several characteristics that distinguish it from traditional EVMS:

- Plan is driven by product quality requirements, not work requirements.
- Focuses on technical maturity and quality, in addition to work.
- Focuses on progress toward meeting success criteria of technical reviews.
- Adheres to standards and models for SE, software engineering, and project management.
- Provides smart work package planning.
- Enables insightful variance analysis.
- Ensures a lean and cost-effective approach.
- Enables scalable scope and complexity depending on risk.
- Integrates risk management activities with the performance measurement baseline.
- Integrates risk management outcomes with the Estimate at Completion.

PBEV augments EVMS with four additional principles and 16 additional guidelines.

PBEV Principles

The following are PBEV principles that set it apart from EVMS:

1. **Product Scope and Quality.** Integrate product scope and quality requirements into the performance measurement baseline.
2. **Product Quality Requirements.** Specify performance toward satisfying product quality requirements as a base

measure of earned value.

3. **Risk Management Integration.** Integrate risk management with EVM.
4. **Tailor PBEV.** Tailor the application of PBEV according to the risk.

The first two PBEV principals are discussed below in greater detail.

Product Scope and Quality

The first principle introduces two control elements that distinguish PBEV from EVMS: product scope and product quality requirements. This principle focuses on customer satisfaction, which is based on delivery of a product that meets its quality requirements and is within the cost and schedule objectives. The supplier has business objectives to achieve maximum customer satisfaction and to deliver the product with the best possible cost performance.

Product Quality Requirements

In the context of PBEV, the product scope is defined and bounded in terms of product quality requirements. A product quality requirement is a characteristic of a product that is mandatory in order for the product to meet verified customer needs. The set of product quality requirements becomes the product requirements baseline that is integrated into the performance measurement baseline along with work scope, schedule, and cost objectives.

Product quality is also discussed in EIA 632 (Requirement 10):

a. Identify product metrics, and their expected values, that will affect the quality of the product and provide information of the progress toward satisfying the acquirer and other stakeholder requirements, as well as derived requirements.
b. Compare results against requirements to determine degree of technical requirement satisfaction, progress toward maturity of the system (or portion thereof) being engineered, and variations and variances from requirements.

Measuring Quality

Project management processes require progress reporting at periodic intervals, normally monthly. However, progress toward achieving product quality objectives is not always measurable on a periodic basis. For example, a hardware or software component may require the completion and assembly of many enabling work products such as drawings or coded software modules, before the integrated set of work products may be measured against product quality objectives. Consequently, interim progress measurement is normally against the scheduled completion of intermediate, enabling work products.

The completion criterion for an enabling work product, such as a drawing, is determined by the organization's process quality procedures and standards. Successful peer reviews or testing are often used to determine the completeness of interim work products against process quality procedures.

PBEV provides guidance to measure performance toward achieving a combination of the following:

- Schedule objectives for enabling work products that meet process quality objectives.
- Event-driven quality objectives when the event is the achievement of measurable product quality requirements.

Also, the achievement of significant performance requirements may not be measurable at the component or subcomponent level but may depend on achieving planned TPM or other quality objectives that are measurable at higher levels of the system architecture. Consequently, EV at the work package level may be quantitatively linked to the performance of integrated components at a higher level of the work breakdown structure.

During the system definition stage, and with the evolutionary acquisition approach, the real product quality requirements are not yet known [10]. Consequently, activity accomplishment criteria should be established to determine progress assessment until the early product quality requirements have been determined.

Evolutionary Acquisition

Per the DAG, when a program uses an evolutionary strategy, each development increment should have a specific set of parameters with thresholds and objectives appropriate to the increment (DAG, Section 2.3.2). Within the development increment, trade studies are used to resolve conflicts between operational capabilities, and functional and performance requirements (Section 4.5.6).

PBEV supports evolutionary acquisition because it is based on requirements, both those that are known by the end of a development increment and those that are evolving during the increment. The work products specified in PBEV Guideline 2.2 include trade study data to substantiate that system requirements are achievable.

PBEV Guidelines

The PBEV guidelines are listed in Table 2 with references to their source standards and models.

Application at Northrop Grumman

PBEV began with a series of process improvements at Northrop Grumman Integrated Systems. The company was driven by the need to improve software development measurement. Initial improvements were based on Practical Software and Systems Measurement (PSM) [11]. Examples of performance-based measures for EV from PSM include functional requirements status, component status, test status, and increment content-function.

A previous CrossTalk article, "Practical Software Measurement, Performance-Based Earned Value," discusses lessons learned, the improvement process, and provides examples of the types of measures that were discarded and implemented [12], i.e., the measurement of defects was retained as an indicator of quality and a predictor of final cost and schedule. However, various measures of achieved requirements were used for schedule progress and EV instead of defect removal. Also provided is advice regarding the suitability of measuring source lines of code, defect and rework planning, and accounting for deferred functionality. Many of these techniques have been incorporated into the NAVAIR handbook, "Using Software Metrics & Measurement for Earned Value Toolkit" [13].

These improvements paid off during upgrades of the B-2 weapon system. The new measures helped to make it a very successful program.

> The B-2 Spirit Stealth Bomber Program implemented several innovative process improvements using EVM. These include integrating earned value with systems engineering processes, defining improved software engineering metrics to support EVM, and developing a leaner, more effective methodology called Performance-Based Earned Value [PBEV]. The PBEV methodology was used to ensure that the warfighter received the most functionality from software development efforts. On Joint Standoff Weapon/Generic Weapon Interface System, we provided 85 percent more capability than originally planned, on schedule and under budget. [14]

Process improvement at the sector is ongoing. Current policy requires alignment of sector processes with IEEE 1220 and a

Performance-Based Earned Value Guidelines	Source	Section Number
1.1 Establish product quality requirements and allocate these to product components.	CMMI ® PMBOK Guide	RD SP 2.1, 2.2 8.1.1.3
1.2 Maintain bidirectional traceability of product and product component quality requirements among the project plans, work packages, planning packages, and work products.	CMMI PMBOK Guide	RM SP 1.4 5.5
1.3 Identify changes that need to be made to the project plans, work packages, planning packages, and work products resulting from changes to the products quality requirements.	CMMI PMBOK Guide	RM SP 1.5 4.3, 5
2.1 Define the information need and objective to measure progress toward satisfying product quality requirements.	CMMI IEEE 1220 EIA 632 PMBOK Guide	MA SP 1.1 6.8.1.5, 6.8.6 4.2.1, 4.2.2 5.2.3.1, 5.5, 8.1.3.5
2.2 Specify work products and performance-based measures of progress for satisfying product quality requirements as base measures of earned value. Examples are the following: • Results of trade-off analysis. • Allocated requirements developed, implemented into design, or tested successfully. • Achieving planned TPMs. • Meeting entry and success criteria for technical reviews. • Other quality objectives achieved.	CMMI CMMI CMMI IEEE 1220 EIA 632 PMBOK Guide	MA SP 1.2 RD SP 3.3 DAR SP 1.5 6.1.1.13, 6.7.6 6.8.1.5, 6.8.6 4.2.1, 4.2.2, 4.5.1 5.2.3.1, 8.2.1.4, 8.1.3.5, 10.3.1.5, Glossary
2.3 Specify operational definitions for the base measures of earned value, stated in precise, unambiguous terms that address: • Communication: What has been measured, how was it measured, what are the units of measure, and what has been included or excluded? • Repeatability: Can the measurement be repeated given the same definition to get the same results?	CMMI PMBOK Guide	MA SP 1.2 8.1.3.2
2.4 Identify event-based success criteria for technical reviews that include development maturity to date and the product's ability to satisfy product quality requirements.	IEEE 1220 EIA 632	3.1.1.6, 4.12, 5.2.4, 5.3.4, 6.4, 6.6, 6.8.1.5 4.2.2
2.5 Establish time-phased planned values for measures of progress toward meeting product quality requirements, dates of frequency for checking progress, and dates when full conformance will be met.	IEEE 1220 EIA 632 PMBOK Guide	6.8.1.5, 6.8.6, 4.2.1, 4.2.2, Glossary 11.6.2.4
2.6 Allocate budget in discrete work packages to measures of progress toward meeting product quality requirements.	IEEE 1220 EIA 632 PMBOK Guide	6.8.1.5, 6.8.6 4.2.1 5.2.3.1, 10.3.1.5
2.7 Compare the amount of planned budget and the amount of budget earned for achieving progress toward meeting product quality requirements.	IEEE 1220 EIA 632 PMBOK Guide	6.8.1.5, 6.8.6 4.2.2, 6.1.2.6 11.6.2.3
2.8 Use Level of Effort method to plan work that is measurable, but is not a measure of progress toward satisfying product quality requirements, final cost objectives, or final schedule objectives.	CMMI LL	MA SP 1.2
2.9 Perform more effective variance analysis by segregating discrete effort from Level of Effort.	LL	
3.1 Identify changes that need to be made to the project plans, work packages, planning packages, and work products resulting from responses to risks.	PMBOK Guide	11.1.3, 11.6.3.2
3.2 Develop revised estimates of costs at completion based on risk quantification.	PMBOK Guide	7.3.2.3
4.1 Apply PBEV coverage to the whole work breakdown structure or just to the higher risk components.	CMMI LL	MA SP 1.2
4.2 Apply PBEV throughout the whole system development life cycle or initiate after requirements development.	CMMI LL	MA SP 1.2

Key to Abbreviations

RD: Requirements Development Process Area
RM: Requirements Management Process Area
DAR: Decision Analysis and Resolution Process Area
SP: Specific Practice
MA: Measurement and Analysis Process Area
LL: Author's Lessons Learned and Process Improvements

Table 2: *PBEV Guidelines*

process architecture that is CMMI-compliant.

Agile Methods

PMs have begun to use agile development methods to streamline the acquisition process. Alistair Cockburn stated that being agile is a declaration of prioritizing for project maneuverability with respect to shifting requirements, shifting technology, and a shifting understanding of the situation [15]. He also discusses an agile approach to using earned value with burn-down charts where the requirements change frequently [16].

However, using agile acquisition streamlining does not justify the elimination of key program documents and solid program planning. Blaise Durante, the U.S. Air Force deputy assistant secretary for Acquisition Integration, stated that implementing Agile Acquisition requires the following [17]:

- Using innovative thought.
- Flexibility.
- Focusing on outcomes versus non-value-added processes and reviews.
- Empowering program managers to use the system versus being hampered by over-staff management.
- Going back to the basics of program management.

PBEV can support agile systems development. Because it uses requirements-based planning and performance-based measurement, it enables innovation, flexibility, and focusing on outcomes instead of non-value-adding processes. Also, PBEV Guidelines 4.1 and 4.2 support agility by tailoring the application of PBEV. Discrete measurement may be applied only to the higher risk components of the WBS and may be deferred until the initial requirements have been developed.

Conclusions

PBEV supplements traditional EVMS with the best practices of SE, software engineering, and project management standards and models. Its principles and guidelines enable true integration of project cost, schedule, and technical performance.

The distinguishing feature of PBEV is its focus on the customer requirements. Measures of product scope and product quality are incorporated into the project plan. Progress is measured against a plan to fulfill all customer requirements. Measuring the wrong things does not dilute management attention. Consequently, management is able to take rapid corrective actions on deviations that threaten customer satisfaction and business enterprise objectives. PBEV also integrates risk management with EVM. Finally, because it is scalable, risk-based, and responsive to changing customer requirements, PBEV can support evolutionary acquisition and agile systems development.

It is recommended that process improvement programs include plans to incorporate PBEV principles and guidelines.◆

References

1. American National Standards Institute. "Earned Value Management Systems." (ANSI)/EIA-748-A-1998. Apr. 1998. Reaffirmed 28 Aug. 2002.
2. Solomon, Paul J. "Integrating Systems Engineering With Earned Value Management." Defense AT&L May/June 2004:42 <www.dau.mil/pubs/dam/05_06_2004/sol-mj04.pdf>.
3. Pracchia, Lisa. "The AV-8B Team Learns Synergy of EVM and TSP Accelerates Software Process Improvement." CrossTalk Jan. 2004: 20-22. <www.stsc.hill.af.mil/crosstalk/2004/01/0401Pracchia.html>.
4. Wynne, Michael. "Policy for Systems Engineering in DoD." Memorandum. 20 Feb. 2004.
5. Department of Defense. Systems Engineering Plan (SEP) Preparation Guide. Ver. 0.95. Washington: DoD, 22 Dec. 2004.
6. Solomon, Paul J. "Practical Software Measurement, Performance-Based Earned Value." CrossTalk Sept. 2001: 26 <www.stsc.hill.af.mil/crosstalk/2001/09/solomon.html>.
7. Project Management Institute. A Guide to the Project Management Body of Knowledge (PMBOK Guide). Newton Square, PA: PMI, 1996 <www.pmibookstore.org/productdetail.asp?productid=4106>.
8. CMMI Product Team. "Capability Maturity Model Integration-Systems Engineering/Software Engineering/Integrated Product and Process Development, Ver. 1.1." Pittsburgh, PA: Software Engineering Institute, Dec. 2001.
9. Solomon, Paul J. "Using CMMI to Improve Earned Value Management." CMU/SEI-2002-TN-016. Pittsburgh, PA: Software Engineering Institute, Oct. 2002. <www.sei.cmu.edu/pub/documents/02.reports/pdf/02tn016.pdf>.
10. Young, Ralph R. Effective Requirements Practices. Addison-Wesley, Mar. 2001.
11. Department of Defense. "Practical Software and Systems Measurement." Ver. 4.0b. Washington: DoD and U.S. Army <www.psmsc.com>.
12. Solomon, "Practical Software Measurement."
13. NAVAIR Handbook. "Using Software Metrics and Measurement for Earned Value Toolkit." Washington, U.S. Navy, Oct. 2004.
14. Solomon, "Practical Software Measurement," 29.
15. Cockburn, Alistair. "Learning From Agile Software Development - Part 1." CrossTalk Oct. 2002 <www.stsc.hill.af.mil/crosstalk/2002/10/cockburn.html>.
16. Cockburn, Alistair. Crystal Clear. Addison-Wesley, Oct. 2004.
17. Durante, Blaise. "Agile Acquisition-Acquisition Streamlining - No Substitute for Solid Program Planning." Agile Acquisition Aug./Sept. 2004 <www.safaq.hq.af.mil/news/aug-sep2004/acq_streamlining.html>.

Note

1. Standards cited include the Standard for Application and Management of the Systems Engineering Process (IEEE 1220), and the Processes for Engineering a System (EIA 632).

About the Author

Paul J. Solomon manages the Earned Value Management Systems (EVMS) for Northrop Grumman Corporation, Integrated Systems. He is an author of the EVMS standard, and received the Department of Defense David Packard Excellence in Acquisition Award. While a Visiting Scientist at the Software Engineering Institute, he authored "Using CMMI to Improve EVM." His book, "Performance-Based Earned Value" will be published by the Institute of Electrical and Electronics Engineers Computer Society. Solomon is a Project Management Professional. He has a Bachelor of Arts and Master of Business Administration from Dartmouth College.

Northrop Grumman Integrated Systems
One Hornet WAY
TD21/2C
El Segundo, CA 90245
Phone: (310) 335-3308
E-mail: solomonpbev@msn.com

Balanced Scorecards: From Golf to Business

Bill Ravensberg
Quality Assurance Analyst

Metrics are all around us and we use them every day – even if we don't realize it. Golf, that frustrating game that we love to play, is used as an analogy to help better relate to the concepts and values of a Balance Scorecard; or, if you understand Balanced Scorecards, then to golf! Any metric is fine by itself but can become much more meaningful and useful when combined with others. Learn why a Balanced Scorecard is an appropriate way of reporting on a collection of metrics. The successful implementation and application of scorecards enables the use of appropriate metrics to facilitate understanding, planning, and communications.

Winston Churchill once noted, "Golf is a game whose aim is to hit a very small ball into an even smaller hole, with weapons singularly ill-designed for the purpose [1]."

Given that, you would wonder why anyone would want to play such a game. Yet, there is an attraction to golf for many people, including me. And, not only do we play the game, but we also do the following:

- Keep our score and compare it to a target known as par.
- Keep track of our scores and establish a personal benchmark known as a handicap.
- Compare our new scores to the benchmark and adjust it when necessary.
- Analyze what we did right.
- Analyze what we could do to improve.
- Like to talk about it … usually!
- Do benchmark studies when we compare our scores and handicaps with others and categorize ourselves in groupings and rankings.
- Chart our progress as well as how we are doing compared to others.
- Create sub-measures such as putts per round, sand saves, driving accuracy, and driving distance to help us understand our strengths and weaknesses and to identify and prioritize where we need to improve.

If we are really good, we can take our statistics and records to businesses to solicit them as sponsors, because they will want to support us and be associated with us when we go on *The Tour*.

And, if measuring ourselves is not enough, the golf equipment manufacturers have done a lot of measurements on their equipment. They have applied their goals and targets to research and development to create the technical advances in today's equipment that golfers enjoy and benefit from. In turn, the manufacturer advertises how successful players have been using their equipment.

Before the Balanced Scorecard

More than 25 years ago, I wanted to get serious about improving my golf game. Basically, I did what many have done. I talked with my buddies and tried to figure out what I should do.

I found that I needed to understand where I was losing strokes. How many putts was I taking on each green? How many penalty strokes was I taking, and why was I taking them? Was I saving or wasting shots around the greens and from the bunkers?

> *"Many people generally look at measures in isolation. The whole picture is not always taken into consideration. But measurement, in isolation, is taking a chance that the results will even be realized."*

I had to pay attention to what I was doing when I played. I made mental notes and compared the results from game to game. I soon learned that the strategy I needed was not on any one specific thing but it needed to be several things together. I now needed to determine how I would go about improving the various parts of my game. I started with the areas I perceived to be the worst and tried to focus on each of them. Unfortunately, perception is not always a great way to go. Neither is trying to work on several different areas at the same time – at least without a strategy. Fortunately, I usually played with the same guys and they were able to help me understand where I needed to improve. Having somewhat analyzed my past performances, I now needed to determine what I was going to do to improve these areas of my game. Especially if I wanted to beat my friends!

Unfortunately, I was not in a position to significantly increase my golf expenses. So, the option of taking lessons could not be considered. I did subscribe to a golf magazine whose format and content I liked. I was also able to get some tips from my golf buddies as we each knew some different things about the game. One of them was a fairly accomplished golfer, and I was able to get a lot of good tips from him.

Next, I needed to practice what I was learning. This involved some work at the driving range and trying some things as I played. I figured that I would mess up every now and then anyway, so it would not matter much if I messed up trying something new. (Note: I do not recommend taking this approach at work!)

Ultimately, I did improve. However, it was a lot of trial and error based on my perceptions of what I was actually experiencing and doing as I played each game.

Like Churchill's thoughts on golf clubs being *ill-designed tools*, the tools and processes I used were ill-designed. I wish there had been a tool that I could have used. Taking a strategic approach to tracking the data and reporting it would have made perfect sense.

What about the tools we use in our work? Are they appropriately designed for use in achieving the desired results?

The Balanced Scorecard - A Useful Tool

According to BetterManagement.com:

> A Scorecard is essentially a carefully selected set of measures derived from an organization's strategy. It's a tool for leaders to communicate to

> employees and external stakeholders the outcomes and performance drivers by which the organization will achieve its strategic objectives. Therefore, the Scorecard provides the link that translates strategy into action across the enterprise, aligning long-term strategy with coordinated, cohesive business activities. [2]

According to *IT Management*,

> The balanced scorecard approach was developed by Dr. David Norton and Dr. Robert Kaplan of Harvard University around 1990. Under this approach, conventional financial measures are augmented by additional measures that report on the learning and growth perspective, and the financial perspective. However, because companies and products vary, one of the challenges of using the balanced scorecard approach is selecting the appropriate metrics for each of the four segments. [3]

The Balanced Scorecard contains four segments: financial perspective, internal business process perspective, customer perspective, and innovation and learning perspective. The *financial perspective* contains the traditional financial measures. Its underlying mission is to represent positive financial contributions by the division to achieving our clients' business goals. Such measures as the average cost to produce a unit of product or process can be used to determine the relative value of the contributions by the division or organization.

The *internal business process segment perspective* contains measures of how the internal processes are performing. The mission is to deliver timely and effective services. We need to know what services and processes, internal to our division, we must excel at to satisfy our customers.

The *customer perspective* contains measures pertaining to things that concern the customer. The underlying mission is to represent how the division is doing in areas that directly affect the clients. Client satisfaction surveys and ratings supporting responses to client queries and problems can be effective in demonstrating customer support.

The *innovation* and *learning perspective* measures the learning and growth of the area. The mission is to develop the internal capabilities to learn, innovate, and exploit future opportunities. Success in this area means we have developed the ability to change and improve, enabling us to better support the customer.

In summary, there are four segments of the Balanced Scorecard, each consisting of a collection of measures. These measures, or metrics, are statistics that we can collect, report, and use to review, evaluate, and determine appropriate action(s).

Table 1: *Sample Balanced Scorecard Using Golf Metrics*

Financial Perspective	Status	Date
Average Cost per Round of Golf	GREEN	August
Number of Golf Rounds per Month	YELLOW	August
Monthly Practice and Learning Cost	GREEN	August
Internal Business Process Perspective	**Status**	**Date**
Average Score	YELLOW	August
Sand Saves	RED	August
Driving Distance	YELLOW	August
Driving Accuracy	RED	August
Greens in Regulation	YELLOW	August
Putts per Greens in Regulation	GREEN	August
Penalty Strokes per Round	YELLOW	August
Customer Perspective	**Status**	**Date**
Golf Buddies' Satisfaction	GREEN	August
Golf Team Satisfaction	GREEN	August
Innovation and Learning Perspective	**Status**	**Date**
Number of Lessons per Month	GREEN	August
Hours per Month Practicing	GREEN	August
Number of New Tips Learned per Month	GREEN	August

GREEN: On the Green   YELLOW: In the Sand   RED: Out of Bounds

When used properly, we make metrics a tool and not the weapon that many have come to fear. Let me demonstrate the use of metrics in a Balanced Scorecard using my golf improvement experience.

Improvement Through Metrics

I wanted to improve my golf scores. I was not happy with what could have been considered my *average score* metric when I started. However, saying that I wanted to improve my score and actually doing it were two different things.

I needed a strategy. I needed to consider several different things as part of my strategy. The sample Balanced Scorecard in Table 1 shows several of the areas mentioned earlier listed as specific measures in the four different segments. It sure would have been useful to have this Balanced Scorecard to report on the state of my golf experience and on each measure by taking all the data tracked from all the rounds of golf played.

Note that in Table 1, your scorecard may have additional measures in a perspective, thus the blank rows. In an automated tool, it is beneficial to have links to the definitions of the metrics, their data charts, and the actual data for the metrics. Also, the frequency of reporting can be whatever is needed to be effective; it does not have to be the same for all measures. Some could be annual, some quarterly, and others monthly.

The status for each item under the four perspectives is determined by comparing the actual results against specific targets. This is how the green, yellow, and red status results are determined. Setting appropriate targets, or goals, is an important requirement. Note that the targets need to be appropriate! Targets help in understanding how the measure is performing in respect to its internal goals and strategies. If you want something like *stretch* objectives, then I would suggest creating a second status.

As you can see, not all the measures in the Internal Business Process Perspective are *on the green*. Some are *in the sand* (yellow status) while others are *out of bounds* (red status). These results matched my game at one time. Using my previously discussed approach, I would have tried to do something in each of these areas to fix my game. However, having data that I can analyze for trends and tendencies can prove quite useful. The data could actually be showing me that the results of the other processes are linked to *driving accuracy*. Perhaps all that is needed is to work on, and improve, the driving accuracy. Then, all the others may inherit better results. I would not be able to determine this if I did not have all the historical data for analysis. This then eliminates any perception that may have resulted in inaccurate and wasted actions.

Another thing to consider is that all the metrics in green status in the Financial Perspective and Innovation and Learning Perspective could be indicating something, too. You may remember that I did not want to increase my financial obligation. But, something the overall metrics are showing is that maybe I needed to reconsider this. Perhaps putting more into lessons and playing more would help bring the process metrics into better shape.

Basically, this demonstrates that there are many variables we need to consider, and that many factors come into play in determining what options can be taken to address improvement. Many people generally look at measures in isolation. The whole picture is not always taken into consideration. But measurement, in isolation, is taking a chance that the results will even be realized. Would you want just one of the subjects in your performance appraisal used? If it is one that you did well in, well, that would be great! But, what if it is a subject you need improving in? And, what if your job or next pay raise was based on it?

As alluded to in Capers Jones' overview on the expanding roles of function point metrics [3], different metrics are appropriate for different companies. We cannot all use the same metrics. We have to evaluate our strategies and then determine what metrics are needed. And, we need to determine what measures work best together to represent a complete picture.

Business, Communication, and Measurement

The business areas of our companies understand measurement. They will do internal and external evaluations and comparisons as a way of measuring and understanding themselves, their products, their competition, and their competitions' products. They will determine the differences and translate their findings into improvements in efficiency and effectiveness to help gain market share.

In turn, the business wants to see an efficient and effective Information Services (IS) division. When a competitor comes out with a new product, our business needs to be responsive to market changes. The IS division, as a service provider to the business areas and company, needs to be accountable and responsive so that these needs can be quickly achieved.

The division needs to be able to communicate in a value-added way with the business. Clarified terms and applications of those terms is a good start. A measurement program can supplement and enhance communication, not only with the business, but also within the division. It is important to ensure consistency within each division and across the organization to ensure the measures contain the same kinds of data so that we can have an *apples-to-apples* comparison and roll up to a multi-division strategic view of all the data.

To be effective in our communications, we need to measure and report actual performance. Do not make things up. Be honest. We need to demonstrate to business management that the IS division is managed with a fact-based approach. The joint evaluation of performance trends versus established goals or targets is an objective and non-emotional way to evaluate and communicate.

Measurement is key and it needs to be relative. It adds meaning and value when looked at on the whole and not individually. It must supply useful information for decision-making. Without measurement, how will we ever know if we are improving our processes and deliverable?

You may want to consider using multiple scorecards. For example, one for development, one for support, one for project management, and one for infrastructure support.

> The Balanced Scorecard complements financial measures of past performance with measures of the drivers of future performance, say Kaplan and Norton. Of course, the old dictum still holds overall: You cannot control what you cannot measure. Thus the Balanced Scorecard was a new concept mainly designed to translate a company's vision and actions into a consistent set of measures. [4]

Effective measurement provides key learning opportunities that can be used in our efforts toward continual improvement. We can use the findings to identify successful practices and build on them. We can eliminate unsuccessful practices and improve those that just are not working the way they were intended. A better understanding and appreciation of other divisions can also be achieved as the development of the metrics and their required data are worked through, reported on, and acted on.

Measurement Lessons Learned

The strategies supporting the mission and vision need to be considered to gain an understanding of what is needed to perform the measurements. This will help in determining the data required to fulfill the measurements.

There are many sources of data when you are dealing with a large number of metrics. Some of the sources include accounting, projects, function point analysis, and surveys. Some data such as function points are not used by themselves for any one measure but are a component of several measures. Other data often used with function points include cost, full-time-equivalent effort, elapsed weeks, and defects.

Ensure the metric definitions are complete. Fully express each definition, including the required data source(s), desired trends, and the rationale to ensure understanding. Even with complete definitions, the data source(s) and formula(s) may change from the original understanding to achieve the definitions' rationale as the metric is developed. Involve the people needed for providing and using the data early in the process to get buy-in. This greatly reduces the time required to *sell* the metrics.

Over time, the current set of metrics may no longer meet the needs. All the different metrics that make up a scorecard need to be monitored and adjusted as strategic plans change. The measurements need to be current and useful. More, or different, metrics may be needed. Therefore, a regular evaluation of the metrics is required.

Data from outside the organization, known as industry benchmarks, can be used to gain an understanding of the performance in respect to other companies in the industry. This data needs to be similar to your own data to ensure an *apples-to-apples* comparison. There are different vendors that have data for this use. They need to be evaluated to determine which one will be an appropriate source of data for your needs.

So, Who Is No. 1?

Like in golf and its manufacturers and players, the IS division can realize improvements and higher satisfaction both internally and with the customer. The scorecard facilitates communication within the division and with the business organization by providing balanced measures with supporting data.

In the July 2003 issue of *Golf Magazine*, Greg Norman commented about how the business world fascinates him. He said:

> That is a wonderful thing that golf

has given me. What amazes me is that some of these businessmen have told me, 'Greg, you know what, we're great in our business, but you have something 99.9 percent of the people in this world don't have. You know what it's like to be number one.' They may be great CEOs, but they don't know if they're the best CEO because it can't be measured. [5]

The Balanced Scorecard can be a very useful tool … when properly used!◆

References

1. Churchill, Winston. QuoteDB.com 11 May 2005 <www.quotedb.com/quotes/2455>.
2. BetterManagement.com. SAS Institute Inc. 11 May 2005 <www.bettermanagement.com>.
3. International Function Point Users Group (IFPUG). IT Measurement: Practical Advice from the Experts. 1st ed. Addison-Wesley Professional, 17 Apr. 2002: 18.
4. IFPUG, 477.
5. Frank, James A. "Golf Talk with Peter Kessler." Golf Magazine July 2003 <www.golfonline.com/golfonline/features/features/article/0,17742,486703,00.html>.

About the Author

Bill Ravensberg, Certified Function Point Specialist, is a quality assurance analyst with a leading Canadian financial institution. He holds the Certified Function Point Specialist designation from the International Function Point Users Group. Ravensberg has been working with Balanced Scorecards since 2001.

409 Brock ST
London, Ontario
N6K ZM3
E-mail: b_ravensberg@hotmail.com

Web Sites

Oklahoma City-Air Logistics Center

www.bringittotinker.com

The 76th Software Maintenance Group at the Oklahoma City - Air Logistics Center is a leader in the avionics software industry that understands the importance of total system integration. The center has a proven track record of producing software on time, on budget, and defect-free. Its staff of software professionals and industry partners provides the expertise, software, weapons, interface, and aircraft systems that are fully integrated to ensure dependable war-winning capabilities. The center's areas of expertise include navigation, radar, weapons and system integration, systems engineering, operational flight software, and more.

Ogden-Air Logistics Center

www.mas.hill.af.mil

The 309th Software Maintenance Group at the Ogden-Air Logistics Center is a recognized world leader in cradle-to-grave systems support, encompassing hardware engineering, software engineering, systems engineering, data management, consulting, and much more. The division is a Software Engineering Institute Software Capability Maturity Model® (CMM®) Level 5 Organization with Team Software Process℠ engineers. Currently the division is transitioning to CMM Integration℠, which integrates systems engineering practices with software engineering processes. This model more closely matches the complex hardware, software, and systems products and capabilities representative of the organization's breadth of products and services.

Warner Robins-Air Logistics Center

https://wwwmil.robins.af.mil

The 402d Software Maintenance Group at the Warner Robins Air Logistics Center provides combat-ready weapon systems, equipment, services, and support personnel for the U.S. Air Force. The center is a leader in systems engineering; reliability, maintainability, and availability engineering; safety engineering; human factors engineering; advanced design and manufacturing engineering; and logistics engineering support. The center has worldwide management and engineering responsibility for the repair, modification and overhaul of the F-15 Eagle, C-130 Hercules, C-141 Starlifter, C-5 cargo aircraft, U-2 surveillance aircraft, all Air Force missiles, all Air Force helicopters, and more.

Bayonets and Deployment

Ever since I was old enough to read, I have been fascinated by the Civil War. As a military dependent growing up, I was lucky to travel a lot, which allowed me to continually talk my father into taking detours to see Civil War sites. While I was actually born outside of the United States (my dad was stationed in Edinburgh, Scotland after World War II), my parents are from Georgia. Travels in the South gave me a very Southern perspective on the "War of Northern Aggression." I was lucky enough to visit Andersonville, Ga., site of the Civil War prison; the battlefields around Richmond, Va.; and Chattanooga, Tenn., site of Lookout Mountain and the "Battle Above the Clouds."

Although I am an avowed Southerner, one of my heroes has always been Joshua Chamberlain, who was an academic, but had a strong urge to fight to save the Union. He volunteered to fight, and was soon made a lieutenant colonel. He fought in many battles and by July 1863, he was made a colonel. On July 2, 1863, Col. Chamberlain was at Gettysburg and was given orders to defend Little Round Top, an important position giving a commanding view of the entire battlefield. His actions during this key engagement held the Union's position and significantly contributed to the Union victory at Gettysburg, a battle that is now viewed as the turning point of the Civil War.

Several years ago, I finally got a chance to visit Gettysburg and made it a point to park my car near a rocky area called Devils Den and walk through the Slaughter Pen up towards Little Round Top. On this area, Col. Chamberlain, running low on ammunition, ordered his men of the 20th Maine to attach bayonets; he led a bayonet charge downhill against the 15th Alabama, driving them back. As a result of this and other heroic actions, Col. Chamberlain (who would be Brevet Maj. Gen. Chamberlain by the war's end) received the Medal of Honor. When the Civil War ended, Gen. Grant chose Chamberlain to receive the formal surrender of weapons and colors on April 12, 1865.

How in the world does this fit in with this issue's theme, "Systems: Fielding Capabilities?" Col. Chamberlain was able to surprise the 15th Alabama with a bayonet charge, and this element of surprise allowed them to succeed. The title of this journal is "CrossTalk, The Journal of Defense Software Engineering." Notice the software engineering. Many times, members of the software engineering profession are a bit shortsighted about their products. It's not the software that will win the war – it's the systems capability.

Being the best shot in the world will not help you if you run low on ammunition. However, knowing that a backup capability exists – and having the training to deploy and use it – will provide you with a winning capability. Under heavy fire, running low on ammo, Col. Chamberlain was able to remember that a backup capability existed. His soldiers had the training to use the backup system, and thus win the battle.

It's the same way in the software world: we need to be able to meet the entire needs of our users, not just stop with producing stovepipe software products that are unable to adapt, integrate, and survive. As software engineers, we need to remember that our job is to help *win* the war, not just produce the software.

Winning is not just about producing a workable software system. It's about meeting and fielding complex systems that have the capability to meet the total needs of our users, including contingency and emergency conditions. It's about the management needed to see potential conditions and prepare software systems and capabilities that meet all of our users' needs.

There should be no such thing as an "unexpected need" at the software level. Requirements engineering needs to be accomplished on two levels – system requirements and software requirements. It is critical to understand the entire system requirements prior to starting software requirements.

In many cases, software design and coding starts prior to complete software requirements. It's not the "correct" thing to do, but sometimes, it's the only choice you have. However, it is absolutely critical to complete system requirements prior to software design and coding. Without complete system requirements, you can't envision the role that the software is going to fulfill in the complete system. If the conceptual design or "vision" of the system is incomplete then the role of software will probably be incomplete as well.

The path to fielding successful capabilities is to make sure that system requirements are fully thought out before assigning system capabilities to software components. This requires planning and analysis. System architects need to be motivated to uncover "elements of surprise".

Perhaps a little prodding with a bayonet would provide useful motivation. I've certainly considered it for a few select co-workers in the past!

— David A. Cook, Ph.D.
Senior Research Scientist
The AEgis Technologies Group, Inc.
dcook@aegistg.com

Can You BackTalk?

Here is your chance to make your point, even if it is a bit tongue-in-cheek, without your boss censoring your writing. In addition to accepting articles that relate to software engineering for publication in CrossTalk, we also accept articles for the BackTalk column. BackTalk articles should provide a concise, clever, humorous, and insightful perspective on the software engineering profession or industry or a portion of it. Your BackTalk article should be entertaining and clever or original in concept, design, or delivery. The length should not exceed 750 words.

For a complete author's packet detailing how to submit your BackTalk article, visit our Web site at <www.stsc.hill.af.mil>.

MAS
Software Engineering Division
Ogden Air Logistics Center

Co-Sponsored by
U.S. Air Force
Air Logistics Centers
MAS Software Divisions

CrossTalk / 309 SMXG
6022 Fir AVE
BLDG 1238
Hill AFB, UT 84056-5820

PRSRT STD
U.S. POSTAGE PAID
Albuquerque, NM
Permit 737